# Staff Summary Sheet

| | To | Action | Signature (Surname), Grade, Date | | To | Action | Signature (Surname), Grade, Date |
|---|---|---|---|---|---|---|---|
| 1 | DFEM | Approve | [illegible], O-6, 30 MAY | 6 | | | |
| 2 | DFER | Review | Kraus, Col 3 JUN 13 | 7 | | | |
| 3 | DFEM | Action | | 8 | | | |
| 4 | | | | 9 | | | |
| 5 | | | | 10 | | | |

| | | | |
|---|---|---|---|
| **Grade and Surname of Action Officer** AD-24/Jensen | **Symbol** DFEM | **Phone** 333-7946 | **Suspense Date** 19 June 2013 |
| **Subject** Clearance of Material for Public Release Case Number: To be Assigned | | | **SSS Date** 30 May 2013 |

**Summary**

USAFA-DF-PA-373

**1. Purpose**: To provide security and policy review on the document at Tab 1 prior to release to the public.

**2. Background**:

- *Author(s)*: Junhua, L., Zhang, Y., Ruths, J., Moreno, D., Wood, K., (Singapore Univ. of Technology and Design) and Jensen, D., (United States Air Force Academy)

- *TITLE*: Innovations in Software Engineering Education: An Experimental Study of Integrating Active Learning and Design-based Learning

- *Abstract*:

Significant advancements have been made in engineering education in recent years. An important outcome of these advancements is the integration and extension of fundamental pedagogies as part of engineering curricula, as well as the need for continued research into the effectiveness of these pedagogies on students' learning within engineering knowledge domains. In this paper, we focus on an engineering educational research study in the domain of software engineering. This study considers the important research question of the efficacy of traditional lecture-homework-project teaching approaches compared to peer-to-peer active learning when combined with design-based learning approaches. The focus of the study is on participants' qualitative assessment and self-reported motivation for learning fundamental programming and software engineering principles when comparing the teaching approaches. The participants are divided into a control group focusing on a traditional teaching approach and an experimental group where the teaching includes a unique integration of peer-to-peer learning and design-based pedagogy. Results from this study demonstrate that software engineering concepts taught in either pedagogical framework are well received by the students, and both groups show distinct improved performance relative to entering the course. However, a peer-to-peer active environment and design-based learning framework are received with much greater interest, engagement, sense of relevance, and intrinsic motivation, especially related to particular classroom and lab experiences.

- *Release information*: Standard release of a USAFA technical report.

- *Previous clearance information*: N/A.

- *Recommended distribution statement*: Distribution A, Approved for public release, distribution unlimited.

**3. Discussion**:

To be published in *Proceedings of the Annual Conference- American Society of Engineering Education*

**4. Recommendation**: Sign coord block above indicating document is suitable for public release. Suitability is based solely on the document being unclassified, not jeopardizing DoD interests, and accurately portraying official policy.

Daniel D Jensen

DANIEL D. JENSEN, PhD, USAF
Department of Engineering Mechanics

1 Tab
Article, Innovations in Software Engineering Education: An Experimental Study of Integrating Active Learning and Design-based Learning

# Innovations in Software Engineering Education: An Experimental Study of Integrating Active Learning and Design-based Learning

**Mr. Liu Junhua, Singapore University of Technology and Design**

Junhua is an undergraduate research technician of the International Design Centre (IDC) and pursuing a BE (Engineering Systems and Design) at Singapore University of Technology and Design (SUTD). He received a Diploma in IT from Singapore Polytechnic. Junhua was awarded the IT Youth of 2013 by the Singapore Computer Society.

**Dr. Yue Zhang, Singapore University and Technology and Design**

Yue Zhang is currently an assistant professor at Singapore University of Technology and Design. Before joining SUTD in July 2012, he worked as a postdoctoral research associate in University of Cambridge, UK. Yue Zhang received his DPhil and MSc degrees from University of Oxford, UK, and his BEng degree from Tsinghua University, China. His research interests include natural language processing, machine learning and artificial intelligence, while he has also been interested in pedagogy improvements in engineering education.

**Prof. Justin Ruths, Singapore University of Technology and Design**

Justin Ruths received a B.S. degree from Rice University, an M.S. degree from Columbia University, and a Ph.D. degree in Systems Science and Mathematics from Washington University in Saint Louis. He is currently an Assistant Professor at Singapore University of Technology and Design. His research interests are in the areas of computational optimal control theory and large-scale complex systems.

**Dr. Diana Moreno, Singapore University of Technology and Design (SUTD)**

Dr. Diana Moreno is a Posdoctoral Associate of the MIT-SUTD Graduate Fellows Program, her research focus is in the areas of innovation processes, ideation methods, design-by-analogy, product development, reliability, concurrent engineering, and design for six sigma. Dr. Moreno completed her PhD studies in Engineering Sciences at Tecnológico de Monterrey co-advised with the University of Texas at Austin, she also holds two Master degrees: M.Sc. in Quality Systems and Productivity at Tecnológico de Monterrey; and M.Sc. in Technology and Management of Energy Companies, in a joint program of the Instituto Superior de la Energía (REPSOL YPF) with Universidad de Navarra's Business School (IESE). She received her B.Sc in Industrial Engineering from Pontificia Universidad Javeriana.

**Dr. Daniel D. Jensen, U.S. Air Force Academy**

Dr. Dan Jensen is a Professor of Engineering Mechanics at the U.S. Air Force Academy where he has been since 1997. He received his B.S. (Mechanical Engineering), M.S. (Applied Mechanics) and Ph.D. (Aerospace Engineering Science) from the University of Colorado at Boulder. He has worked for Texas Instruments, Lockheed Martin, NASA, University of the Pacific, Lawrence Berkeley National Lab and MSC Software Corp. His research includes design of Micro Air Vehicles, development of innovative design methodologies and enhancement of engineering education. Dr Jensen has authored approximately 100 papers and has been awarded over $3 million of research grants.

**Prof. Kristin L. Wood, Singapore University of Technology and Design (SUTD)**

Dr. Kristin L. Wood is currently a Professor, Head of Pillar, and co-Director of the International Design Center (IDC) at Singapore University of Technology and Design (SUTD). Dr. Wood completed his M.S. and Ph.D. degrees in Mechanical Engineering (Division of Engineering and Applied Science) at the California Institute of Technology, where he was an AT&T Bell Laboratories Ph.D. Scholar. Dr. Wood was formerly a Professor of Mechanical engineering at the University of Texas (1989-2011), where he established a computational and experimental laboratory for research in engineering design and manufacturing. He was a National Science Foundation Young Investigator, the Cullen Trust for Higher Education Endowed Professor in Engineering and University Distinguished Teaching Professor at The University of Texas at Austin.

# Innovations in Software Engineering Education: An Experimental Study of Integrating Active Learning and Design-based Learning

## ABSTRACT

Significant advancements have been made in engineering education in recent years. An important outcome of these advancements is the integration and extension of fundamental pedagogies as part of engineering curricula, as well as the need for continued research into the effectiveness of these pedagogies on students' learning within engineering knowledge domains. In this paper, we focus on an engineering educational research study in the domain of software engineering. This study considers the important research question of the efficacy of traditional lecture-homework-project teaching approaches compared to peer-to-peer active learning when combined with design-based learning approaches. The focus of the study is on participants' qualitative assessment and self-reported motivation for learning fundamental programming and software engineering principles when comparing the teaching approaches. The participants are divided into a control group focusing on a traditional teaching approach and an experimental group where the teaching includes a unique integration of peer-to-peer learning and design-based pedagogy. Results from this study demonstrate that software engineering concepts taught in either pedagogical framework are well received by the students, and both groups show distinct improved performance relative to entering the course. However, a peer-to-peer active environment and design-based learning framework are received with much greater interest, engagement, sense of relevance, and intrinsic motivation, especially related to particular classroom and lab experiences.

## I. INTRODUCTION

Over the last two decades, significant advancements have been made in the development of active learning approaches in many fields of engineering. These efforts focus on transforming course content from passive, traditional classroom environments to modes where the students take a much more dynamic and participatory role. Similar advancements have been made in project-based learning (PBL) and design-based learning (DBL). In these cases, students learn in an application-based environment, typically in teams, and, in the preferred mode, directed by open-ended problems where no single right answer exists.

While the literature reports a number of examples of active learning and design-based learning approaches within engineering, few investigations are reported regarding their integration. This void especially exists in certain disciplines of engineering, such as software design – typically found in computer engineering, electrical engineering, and computer science programs.

Students in these programs, and in particular Computer Science or Software Engineering courses, are often excited at first, but gradually lose interest and motivation while tackling more and more difficult theoretical problems. As a result, it is extremely important to explore methods to consistently engage students while equipping them with fundamentals, technical skills, innovation skills, and learning skills. Research shows that design-based engineering curriculums are able to intrigue students' curiosity in exploring a new area; while an active learning approach enables consistent engagement to students through probing discussions and activities. Hence, integrating both design-based learning and active learning into Software Engineering curricula could potentially deliver an enriching and engaging learning experience to students where they exhibit increased motivation, perceive greater relevance of course content, and demonstrate high interest in subject matter.

In this paper, we consider a systematic investigation of combining active learning and design-based learning for the instruction of software engineering content. The investigation entails an experiment based on two scenarios: a control group, focusing on the traditional approaches, and an experimental group, focusing on active learning through peer to peer instruction and the integration of design project modules. For the purposes of this study and educational needs of the students, the experimental environment is chosen to be an intermediate-level short course on objective-oriented programming in Java. The control group follows a traditional learning approach, where content is taught in a combination of lectures, labs, and homework assignments. The experimental group, on the other hand, learns content through pre-assigned readings, peer-peer active presentations and discussions of course content, faculty-led follow-up discussions of content at a peer level, connections of provocative and real-life examples to motivate course content (i.e., "show and tell"), and design-based problems integrated throughout classroom and out-of-class activities.

Therefore, present research intends to identify if there is significant difference between two learning approaches through following metrics: 1) participant's pre and post-test performance, and 2) learning approach feedback about engagement, student interest, and motivation for software engineering content.

## II. Related Research and Pedagogical Underpinnings

Engineering education continues to change as we encounter more interdisciplinary learning and create access for a wider variety of students to be strive for success within higher-education classrooms (Adams, *et al.*, 2004; Anderson and Northwood, 2002). In better understanding the intricacies of students' learning, educators can improve the teaching of engineering concepts. A variety of salient educational theories guide the creation, assessment, and improvement of engineering domain content, within an active-learning context. These learning theories range from how information is presented and processed to understanding an individual student's personality and preferences. Here we focus on developments in active learning approaches particularly tailored and found prominently in engineering education curricula and / or research.

### *II.1 Active Learning, Interactive Engagement, & Constructivist Theories*

Active learning approaches improve students' overall learning, a view shared generally by faculty teaching engineering education (Aglan, 1996). There is considerable literature that addresses the advantages of using active learning in STEM curriculum (Aglan, 1996; Bonwell; Dennis, 2001; Eder, 2001; Hsi, 1995; Holzer, 2000; Linsey, 2006, 2007, 2009; Mayer, 2002; Meyer, 1994; Prince, 2004; Stice, 1987; Talley, 2007; Welsh, 2007; Wood, 2000, 2001, 2002, 2004; Barr, 2000; Bean, 2001). These literature sources show that students' motivation and learning are simultaneously enhanced by the incorporation of active learning into the classroom.

The foundation of active learning is a constructivist teaching philosophy and, in particular, social constructivism. Through the interaction with ideas, concepts, materials, and other artifacts, students construct their knowledge. This approach seeks to alter the mode of knowledge and conceptual understanding through student construction as opposed to passive reception. Modules may be framed within a constructivist framework in an effort to create engineering education experiences that (1) help students to construct deeper understanding of theoretical concepts in connection to practical experiences; (2) facilitate students' engineering skills; and (3) develop students' capabilities and dispositions for engaging in collaborative project-based inquiry and critical thinking. To assimilate new information and incorporate it into the existing knowledge, students need to restructure their knowledge for themselves, which can be accomplished through active learning. A number of tenants underlie this teaching philosophy, including (Knight, 2004): students take direct responsibility for their knowledge, proactively engaging in the study of their texts and reference materials, participation and leadership in course activities, completing assignments, laboratories, and exploration in the field; the instructor assumes more of a role of a facilitator: "a guide on the side, not a sage on the stage; students receive immediate feedback and guidance on their work and class contributions; and students invest and spend a significant portion of class time engaged in concrete experiences (doing) and reflections, contemplation and critically thinking, and discussing the topic area of the course. This approach is opposed to a passive listening of the topic from others.

Active learning or interactive engagement does not comprise a single approach but many approaches may be executed through a variety of modes and media. Exemplar delivery modes, with extensive testing (e.g., in the area of STEM physics (Laws, 1997)), include Cooperative Groups (Heller, 1992a, 1992b), Socratic Dialogue Inducing Labs (Hake, 1987, 1992), Interactive Demonstrations (Sokoloff, 1997), Peer Instruction (Mazur, 1997), Think/Pair/Share (Van Heuvelen, 1999), Tutorials (McDermott, 1994, 1998), and Hands-on Activities (Linsey, 2006, 2007, 2009).

Extensive empirical studies have been carried out to understand the role of active learning. Clear evidence exists that the lecture mode of instruction and passive reception by students is not effective at leading to student understanding. Many contributory factors explain this result (Knight, 2004): Students are not good listeners in a passive instructional setting. They simply do not know how to listen. Critical thinking skills cannot be developed or applied to subject matter due to the pace of content presentation. The meaningful attention span of individuals is 10-15 minutes. Many orally delivered lectures simply reiterate material in the text or references for a course or topic area. Lectures typically focus on abstractions not concrete experiences with a subject matter.

In the area of STEM physics, extensive testing between conventional teaching approaches versus active learning modes showed more than a two times (100%) improvement regarding the conceptual understanding of students with a Newtonian Force Concept Inventory. These results are striking and correlated closely with students' abilities to develop and employ good problem solving skills (Hake, 1998). Depending on one's teaching styles, the intent is to integrate interactive engagement throughout innovative initiatives in courses and the cultural setting of the students' educational environment.

## *II.2 Integrated Learning Experiences: the Kolb Cycle*

The Kolb model shown in Fig. 1 describes an entire cycle around which a learning experience progresses. The goal is to structure learning activities that will envelope the entire cycle, providing the maximum opportunity for full comprehension and self-learning. Increasing engineering educators are wielding this model to evaluate and enhance engineering teaching. Software engineering models may be designed to provide learning experiences motivated by the Kolb cycle that are not well met with traditional course instruction. Specifically, software engineering learning modules may be developed with need-based, service learning, or industry or research relevant problems. This provides the "Concrete Experience" component of the cycle in a similar manner as a case study. The "Reflective Observation" part of the cycle may be accomplished by providing key times for student questions, critique, and assessment questions throughout the learning modules. These observational opportunities may be designed to encourage the students to reflect on the innovation history, processes, problem, theoretical frameworks, ideas, and / or decisions. The "Abstract Hypothesis and Conceptualization" component of the Kolb cycle may be addressed through the use of the course content itself, in addition to supporting conceptual course material. A concept inventory may lead the construction and assessment goals of this Kolb component, led by student peer-to-peer interactions. Finally, the "Active Experimentation" part of the cycle may be met when the students will be asked to interact and iteratively solve distinct modules, either reproducing known concepts in the subject matter, or addressing a building block of course concepts.

## *II.3 Design-Based Learning (DBL)*

Significant advancements have been made in design-based learning (DBL), especially in the context of active- and hands-on learning (Sheppard, 1992; Dym, 1994, 1999, 2003-2005; Crawford, 1994; Eder, 1994; Dally, 1993; His, 1995; Dutson, 1997; Otto, 1998, 2001; Barr, 2000; Tavakoli, 2000; Newman, 2001; Dennis, 2001; Jensen, 2000, 2005; Bilen, 2002; Blessing, 2002; Campbell, 2002; Ulrich, 2004; Green, 2004; Charyton, 2009; Linsey, 2010; Markman, 2011; White, 2011; Wood, 2000, 2001, 2002, 2005, 2012). These advancements, in various direct and indirect ways, provide frameworks for learning through open-ended problems, creative problem-solving, and engagement in service-learning and society-based projects. The work reported in this paper builds upon these foundations and advancements.

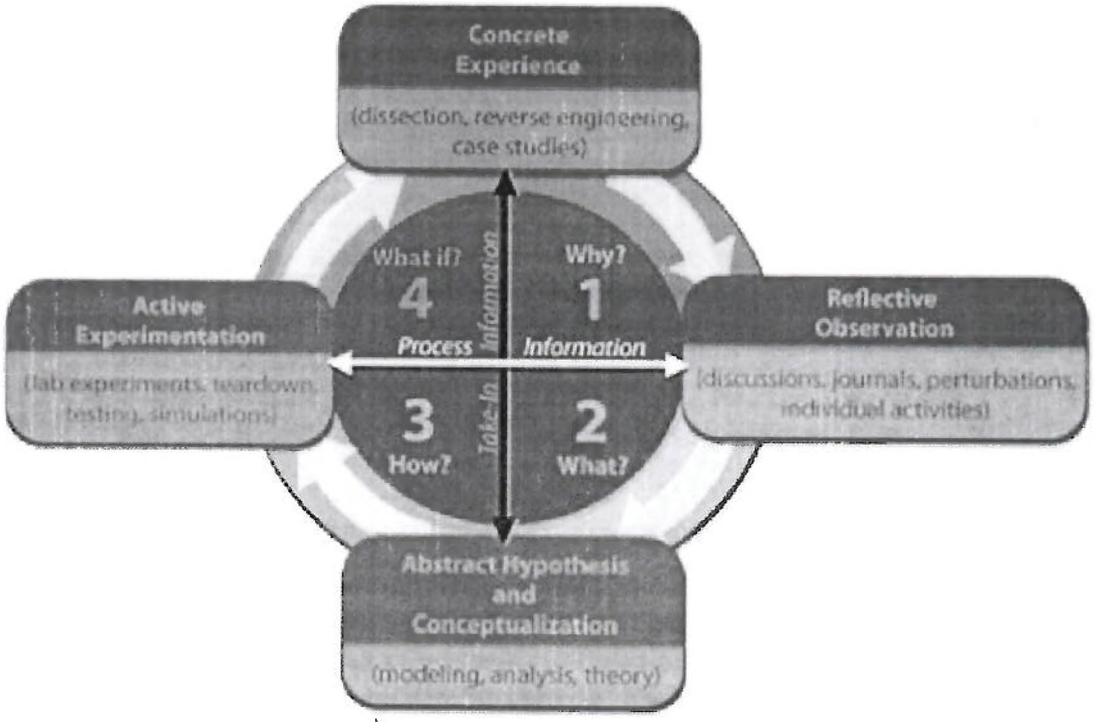

Concrete Experience
(dissection, reverse engineering, case studies)
What if?
4
Process
Information
Why?
1
Information
Active Experimentation
(lab experiments, teardown, testing, simulations)
Reflective Observation
(discussions, journals, perturbations, individual activities)
3
How?
Take-In
2
What?
Abstract Hypothesis and Conceptualization
(modeling, analysis, theory)

Figure 1. Kolb cycle for integrated learning experiences in assimilating and processing information

### *II.4 Performance assessment design*

Performance assessments are used to examine the quality of studies (Messick, 1994). Depending on the nature of the studies, performance assessment can be product-driven or performance-driven. Product-driven assessment evaluates the outcome of the domain whereas performance-driven assessment focuses on the quality of performance in the process. In cases where evaluation of the product or performance is the focus, replicability and generalizability are negligible. In the performance assessment of competencies, performance-driven assessment design has to consider generalizability and replicability in order to measure how consistent the meaning of the issue is likely to be. In this work, we build on Messick's contributions to develop assessment instruments.

## III. Experimental Study Design

Building on the related research and pedagogical underpinnings in Section II, we consider here the design of the experimental study. The primary hypothesis of the research study is as follows: ***"There exists significant improvement in the engagement, student interest, and motivation for software engineering content using an integrated approach of active and deign-based learning compared to traditional teaching approaches."*** Traditional approaches refer to a combination of lectures, tutorials and lab sessions for a software engineering course.

To test this hypothesis, the experimental study included the design of software-engineering course content, coordination of the study's control (traditional) and experimental (active-DBL) groups, and development of the assessment and evaluation instruments.

### *III.1 Course Design*

***Content Framework***. For the purposes of this study, a five-day short course, offered during independent activity periods (IAPs) as vacation period where the students do not take curriculum courses, is chosen as the venue to carry out the experimental study. This venue provides a number of advantages to carry out the study, namely: (i) new courses may be offered with relative ease compared to core curriculum courses or electives; (ii) greater experimental control may be obtained relative to consistency in instructor talents and skill set, environmental conditions across the experiment, and the availability of student time and resources; (3) participants in the study are volunteers, with personal choice and interests as the motivation to enroll in the course; and (4) significant contact hours, in this case approximately seven hours per day for three to five days, exist with the students in a relatively short overall timeframe.

The software engineering topic chosen tor the five-day short course is object-oriented programming (OOP), where the focus language is Java. The learning objectives for the students focus on gaining intermediate mastery of OOP concepts and apply these concepts in a designette project (Wood, 2012). The course concepts are explored during the first three days of the short course, followed by two days of designette project work and presentation of the results. In view of the course timeframe, a balance is sought between breadth and depth of the course content while designing the course framework. While obtaining programming expertise is one of the learning objectives of the course, we expect the students to be familiar with some basic knowledge of Java programming, such as its syntax and the use of Integrated Development Environment (IDE). This expectation is based on the type of students typically interested in such IAP software engineering experiences and validated by the volunteer students that enrolled in the course. In spite of this expectation, instructors still devoted a minimum amount time in reviewing basic background knowledge. This review served as a bridging session from the basics to intermediate level (and even some advanced) OOP concepts.

The main course framework consists of three sections. They are Fundamental Object-Oriented Concepts, Java Libraries, and Integrated Techniques. The Object-Oriented Concepts section seeks the study of Objects, Classes, Encapsulation, Inheritance and Polymorphism. These concepts are crucial for students to design and engineer interactive, effective and reusable codes to build sustainable system architecture.

Java Libraries refers to pragmatic Java's packages and functions that are pre-developed and integrated with the Java Development Kit (JDK). In the designed course, we introduce the basic packages and functions such as the Collection frameworks and Java I/O (Input/Output). Although these libraries belong to the Java language, other object-oriented languages, such as Microsoft's C#, too have similar libraries. As a result, familiarizing with the built-in classes and interfaces is very helpful for software engineers to tackle problems while building on well-known modules.

Last but not least, the Integrated Techniques section introduces the use of Generics and Enumerations. These topics strengthen the flexibility developing software. By acquiring skills with these two techniques, students are able to better tackles uncertainties, such as receiving ambiguous object types in various situations.

Figure 2 shows the organization and breakdown of this course content framework. This framework, while specifically designed for the experimental study and short course, relates and overlaps significantly with semester-level courses on object-oriented programming.

***Concept Inventory***. In creating the course content framework as shown in Fig. 2, we developed a concept inventory for the focus area of object-oriented programming. Concept inventories have been developed in a number of fields, such as mechanics, geosciences, physics, signals and systems, astronomy, statistics, computer engineering (CECI, 2012), and chemistry. To the best of our knowledge, a software or computer engineering concept inventory has not been developed for object-oriented programming.

We systematically developed an object-oriented programming concept inventory to support the course and experimental study design. The concept inventory was formed based on OOP literature and text books in the field. In particular, it was developed with reference to six textbooks we used in concert with the course development. These six textbooks are (a) Thinking in Java Fourth Edition (Eckel, 2003), (2) Head-First Java Second Edition (Sierra, 2005), (3) The Object-Oriented Thought Process Third Edition (Weisfeld, 2009), (4) Java – How To Program Ninth Edition (Deitel, 2012), (5) Effective Java Second Edition (Bloch, 2008), and (6) The Sun-Certified Java Programmer 6.0 (SCJP 6.0 or 310-065) Exam Study Guide (Manning, 2009). Other supporting references include (Winblad, 1990; Budd, 1991; Martin, 1992).

The concept inventory includes key object-oriented theories, principles, and practical problems. It includes both theory and application approaches on each topic. It then formed the basis of designing the course content framework, as shown in Fig. 1, in addition to pre- and post-test assessment and evaluation instruments for the course and study. Though we developed the concept inventory systematically as the intersection of noted key concepts and student conceptual misunderstandings, the concept inventory, shown by Fig. 1 and our pre- and post-test, may be evolved and improved significantly. One needed evolution is to solicit experts across software engineering, and object-oriented programming and design in particular, to collaboratively develop a meaningful and long-lasting inventory. We could collect questions through surveys and long-distance collaborations, gather relevant comments and suggestions, and then systematically verify and validate the resulting concept inventory. The concept inventory developed in this work, as exemplified by Fig.1 and our pre- and post-test, represents a significant step toward a robust concept inventory for OOP.

| | | |
|---|---|---|
| Part 1: Object-Oriented Concepts | **1.1** | **Objects insight** |
| | 1.1.1 | Object consists of States and Behaviors |
| | 1.1.2 | Class consists fields, Methods and Arguments |
| | 1.1.3 | Object is an instance of Class; Class is the blueprint of object. |
| | **1.2** | **Encapsulation** |
| | 1.2.1 | Minimize the accessibility of classes and members (Why encapsulating/hiding data) |
| | 1.2.2 | Getter and Setter |
| | **1.3** | **Inheritance** |
| | 1.3.1 | Is-a vs Has-a relationship (Inheritance vs Composition) |
| | 1.3.2 | Favour composition over inheritance |
| | 1.3.3 | Interfaces vs abstract base types |
| | 1.3.4 | Favour interfaces over abstract base types |
| | 1.3.5 | When and how to create inheritance hierarchy |
| | **1.3** | **Polymorphism** |
| | 1.3.1 | Why Polymorphism - OO design, extensibility |
| | 1.3.2 | User object as its own type and its base type |
| | 1.3.3 | How to construct polymorphism |
| Part 2: Java Libraries | **2.2** | **Containers** |
| | 2.1.1 | Idea of a container |
| | 2.1.2 | Collections |
| | 2.1.3 | Build-in Collections: Set, List, Map |
| | **2.3** | **Java i/o** |
| | 2.3.1 | Object Serialization: syntax and application |
| | 2.3.2 | Java.io.File: buffer, writer and reader |
| | 2.3.3 | Introduction to XML |
| Part 3: Integrated Techniques | **3.1** | **Enum** |
| | 3.1.1 | Introduction to enum: custom value type |
| | 3.1.2 | When and how to use enum |
| | 3.1.3 | Add methods to enum |
| | 3.1.4 | Using enum with switch |
| | **3.2** | **Generics** |
| | 3.2.1 | Why generics - containers that hold multiple types |
| | 3.2.2 | How to construct Generic types |

Figure 2. Object Oriented Programming Course Content Framework for Experiment Study

### *III.2 Coordination: Course and Experimental Study*

***Participants***. The short course was offered to students from a variety of educational and experiential backgrounds. Recruitment included senior students from Singapore Polytechnic, a secondary-level engineering diploma granting institution, and freshmen from Singapore University of Technology and Design (SUTD). Figure 3 lists relevant demographic data regarding the participants. The background of the students from these institutions was viewed to be commensurate because of the engineering diploma emphasis of Singapore Polytechnic. Of course, some age differences exist in the students, which we attempted to control through the distribution of students amongst control and experimental groups.

As the short course was designed to be intermediate level, participation of the course required students to have basic object-oriented programming background or experience in other programming practices. While registering for the course, students completed a pre-survey to indicate their programming background. These data were used, in part, as an indicator to distribute participants across a control group and experimental group for the experimental study.

All students from Singapore Polytechnic completed at least one object oriented programming module as part of their curriculum. Some have experience in other programming modules such as Java programming, data structures, and algorithm and systems development.

All students from SUTD are first freshmen. All except one student had object oriented programming experience in either Java or C++. The remaining student had limited programming experience.

| *Participant* | *Gender* | *Prior Programming Experience* | *Year* | *Institution* |
|---|---|---|---|---|
| 1 | Male | Java, C#, Data Structures and Algorithms | Year 2 | SP |
| 2 | Male | Java, C++, PHP, HTML | Year 1 | SUTD |
| 3 | Male | C++ and Java | Year 1 | SUTD |
| 4 | Male | C++, Objective C, Cocoa,Carbon | Year 1 | SUTD |
| 5 | Male | Java, C# and ActionScript | Year 1 | SUTD |
| 6 | Female | Java, JavaScript, ActionScript 3, ASP.NET, C#, SQL, HTML | Year 2 | SP |
| 7 | Male | Java | Year 2 | SP |
| 8 | Male | TI-basic | Year 1 | SUTD |
| 9 | Male | JAVA, Data Structures and Algorithms, C#, ASP.NET, PHP | Year 2 | SP |
| 10 | Male | Limited | Year 1 | SUTD |
| 11 | Male | C++ | Year 1 | SUTD |
| 12 | Male | Java, C, C++. PHP, JSP, action script 3.0 | Year 1 | SUTD |
| 13 | Male | Objective C, Javascript, Java | Year 1 | SUTD |
| 14 | Male | Java, C++, Objective C, Python | Year 1 | SUTD |
| 15 | Male | Octave, Python, Javascript, C | Year 1 | SUTD |
| 16 | Female | Java,Data Structure and algorithm, c#, HTML | Year 2 | SP |

Figure 3. Participants' background and relevant demographics

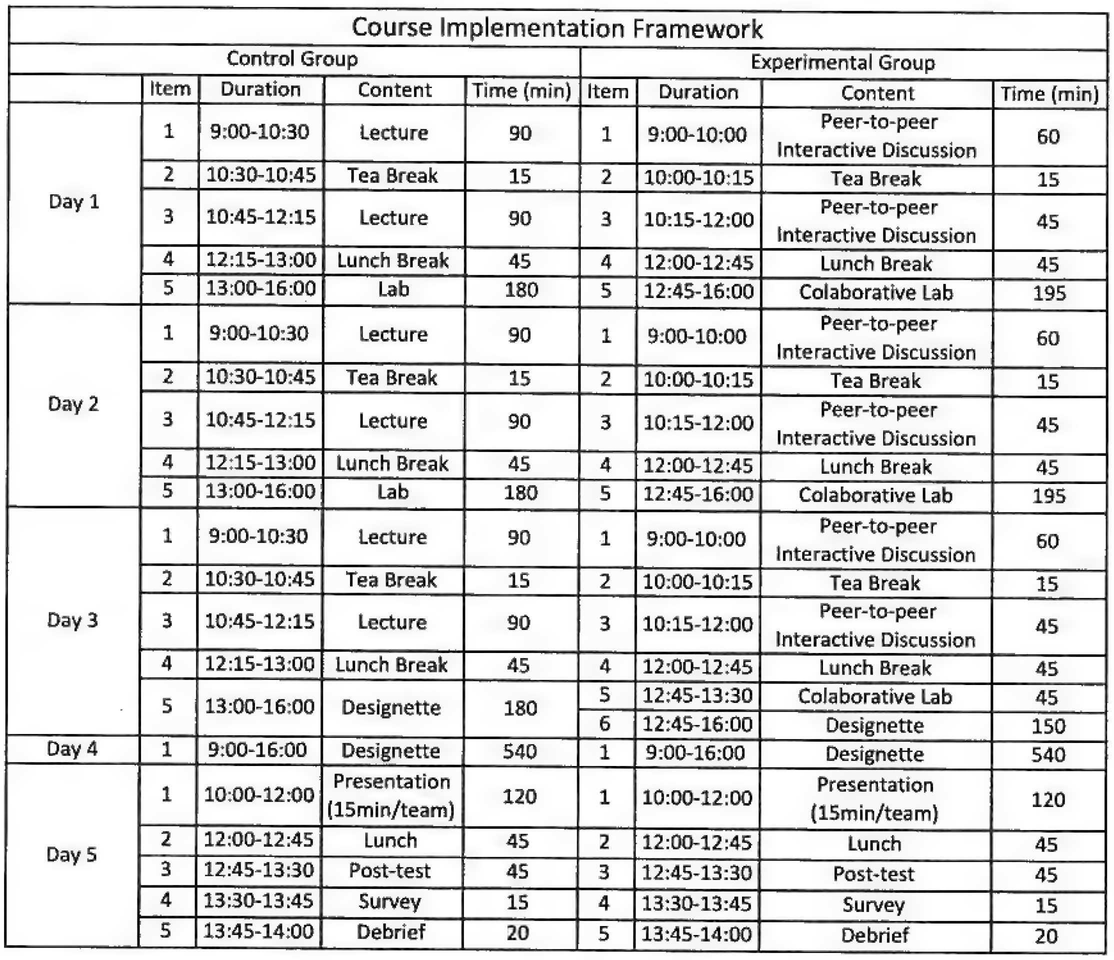
Students that volunteered for enrollment are brought together for a course orientation. The students are told that the short course would include a pedagogical experiment, based on voluntary participation. However, information such as how both classes would be conducted and their curriculums were not disclosed to students.

At the end of the orientation, all students are given a pre-test based on the concept inventory developed as part of the project. Upon completion, the students are then assigned to either the control or experimental groups, where they are evenly distributed according to educational level and skills in software programming based on the data in the pre-survey.

After separating the participants into control and experimental groups, the short course was implemented over a one-week period. The participant groups remained segregated for course content instruction and course exercises until the final day of the course where the students presented results of a final designette project (Wood, 2012). Faculty members teaching the short course were recruited to have similar expertise, enthusiasm, and content knowledge, independent of the control and experimental group structure. They instructors were given the same content material, but were not informed about the pedagogical structure being implemented in the other instructor's group.

***Control Group***: The control group uses a traditional approach in teaching. It includes lectures, lab sessions and homework. On the other hand, the experimental group uses active learning through peer-to-peer instruction and the integration of design project modules. Figure 4 shows the mapping of the course content framework, Fig.1, to both the control group and experimental group pedagogical structures.

The structure for the control group includes a combination of lecture, lab and homework, carried out in five consecutive days. In the morning session of each day, the students receive two 3-hour lectures with a 15-minute break in between. The instructor gives the lectures using whiteboards and markers.



| Course Implementation Framework | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Control Group | | | | Experimental Group | | | |
| | Item | Duration | Content | Time (min) | Item | Duration | Content | Time (min) |
| Day 1 | 1 | 9:00-10:30 | Lecture | 90 | 1 | 9:00-10:00 | Peer-to-peer Interactive Discussion | 60 |
| | 2 | 10:30-10:45 | Tea Break | 15 | 2 | 10:00-10:15 | Tea Break | 15 |
| | 3 | 10:45-12:15 | Lecture | 90 | 3 | 10:15-12:00 | Peer-to-peer Interactive Discussion | 45 |
| | 4 | 12:15-13:00 | Lunch Break | 45 | 4 | 12:00-12:45 | Lunch Break | 45 |
| | 5 | 13:00-16:00 | Lab | 180 | 5 | 12:45-16:00 | Colaborative Lab | 195 |
| Day 2 | 1 | 9:00-10:30 | Lecture | 90 | 1 | 9:00-10:00 | Peer-to-peer Interactive Discussion | 60 |
| | 2 | 10:30-10:45 | Tea Break | 15 | 2 | 10:00-10:15 | Tea Break | 15 |
| | 3 | 10:45-12:15 | Lecture | 90 | 3 | 10:15-12:00 | Peer-to-peer Interactive Discussion | 45 |
| | 4 | 12:15-13:00 | Lunch Break | 45 | 4 | 12:00-12:45 | Lunch Break | 45 |
| | 5 | 13:00-16:00 | Lab | 180 | 5 | 12:45-16:00 | Colaborative Lab | 195 |
| Day 3 | 1 | 9:00-10:30 | Lecture | 90 | 1 | 9:00-10:00 | Peer-to-peer Interactive Discussion | 60 |
| | 2 | 10:30-10:45 | Tea Break | 15 | 2 | 10:00-10:15 | Tea Break | 15 |
| | 3 | 10:45-12:15 | Lecture | 90 | 3 | 10:15-12:00 | Peer-to-peer Interactive Discussion | 45 |
| | 4 | 12:15-13:00 | Lunch Break | 45 | 4 | 12:00-12:45 | Lunch Break | 45 |
| | 5 | 13:00-16:00 | Designette | 180 | 5 | 12:45-13:30 | Colaborative Lab | 45 |
| | | | | | 6 | 12:45-16:00 | Designette | 150 |
| Day 4 | 1 | 9:00-16:00 | Designette | 540 | 1 | 9:00-16:00 | Designette | 540 |
| Day 5 | 1 | 10:00-12:00 | Presentation (15min/team) | 120 | 1 | 10:00-12:00 | Presentation (15min/team) | 120 |
| | 2 | 12:00-12:45 | Lunch | 45 | 2 | 12:00-12:45 | Lunch | 45 |
| | 3 | 12:45-13:30 | Post-test | 45 | 3 | 12:45-13:30 | Post-test | 45 |
| | 4 | 13:30-13:45 | Survey | 15 | 4 | 13:30-13:45 | Survey | 15 |
| | 5 | 13:45-14:00 | Debrief | 20 | 5 | 13:45-14:00 | Debrief | 20 |

Figure 4. Mapping of course content framework (Fig. 1) to course pedagogy (Course Implementation Framework)

In the afternoon session, the students work in the lab, where they are given a problem set selected from "Thinking in Java" (Eckel, 2003), and a reading assignment that is relevant to the topics of the next day. A post-doctoral research associate works in the lab to assist the students and answer questions. The students are expected to complete half to two-thirds of the problems within the 3-hour lab session, leaving the remaining work and reading assignment to be completed as homework.

The contents of the lectures are organized as follows. A general overview of programming is given in the beginning, which includes an introduction of the purpose and essence computer programming, and a brief history of programming languages. Java is identified in a taxonomy of programming languages. A key concept introduced in this lecture is that programming is a means by which humans control computers, and hence the human model and the computer model are two sides of the story about computer programming.

For object-oriented programming, and in particular Java, the human model is object oriented programming (OOP) and the computer model is the Java virtue machine on an abstract computer architecture. The remainder of the lecture series is given by telling both sides of the story, and how Java syntax and concepts are designed to connect them.

One lecture is given on the basic concepts of OOP, including encapsulation, inheritance and polymorphism. An example is used to explain the above concepts, with pseudocode represented in UML. The difference between composition and inheritance is discussed, highlighting the differences and possible design choices. Multiple inheritance is explained after the introduction of single inheritance, and parameterized types are introduced at the end of this lecture.

On the other side of the story, one lecture is given on the basic computer architecture and the Java virtual machine. A computer word is explained after a brief review of binary numbers. Computer architecture is discussed in terms of a CPU and registers, memory, I/O devices and the central bus. A reflection of the first lecture is given by explaining why this architecture can achieve what computers do today, including robotics, multimedia and others. Chip instruction sets were discussed during the introduction of a CPU, and simple instructions of the Intel X86 chipset were reviewed. The organization of a program in the memory is discussed, with the split of the data section into a stack and a heap explained in elaboration.

The remainder of the lectures consists of two main topics, including Java Syntax, OO concepts and libraries. The syntax of Java is introduced in three main sections. The first section includes primitive types and values, expressions, statements, control flow, functions, classes, garbage collection, code compilation, packages and comments. The basic concepts were organized in the sequence of a traditional textbook, so that after all concepts were introduced, the students could write a complete Java program. Special classes such as String and Array are introduced when Classes are introduced. Boxing and Enum are also introduced in this section. Programming concepts are always discussed with their JVM behavior.

The second section consists of access control, inheritance, polymorphism, interface, inner class and generics. These concepts were introduced in correspondence with the Java Syntax introduced previously. For example, interface and inner class are introduced as implementations of multiple inheritances. On the machine side, JVM behavior of each syntax type is discussed. The third section consists of language features including exceptions, reflection and annotations. Again both the design of programming paradigms and the implementation on the Java virtual machine are explained. The second main topic is Java libraries, where containers and file I/O are highlighted.

The philosophical basis for the traditional lecturing includes the laying out of a big picture and consistency in storytelling. The lecturer puts all relevant knowledge under a broad theme, and discusses each topic with regard to the theme. The students have a comparatively uninterrupted reception of the whole flow of concepts from a single consistent point of view.

During our control group lectures, questions were encouraged, and the students were engaged. The lecturer asked relevant questions to the students after the introduction of each concept to make sure that most students understood the concepts. Missing of particular concepts by a small portion of students was deemed as understandable and acceptable by the instructor, while the lecturer always made sure that the students knew the big picture so that they would be encouraged to learn by themselves the missing content, knowing their position in the broad lecture theme.

A potential disadvantage of the control group was that the lectures and lab sessions were given completely separately, without extensive communication between the lecturer and the teaching assistant. For future experiments we would make the lab sessions more closed linked to the lectures by having the lecturer design the lab work, and increasing communication between the lecturer and the RA.

***Experimental Group***. The course framework for the experimental group consists of four main components: pre-reading, peer-to-peer interactive discussion, collaborative lab, and integrated design-based learning designettes (Wood, 2012). Figure 5 shows an experimental group setting where students are paired into teams of peer-to-peer learners, and students initially direct the content in a given session.

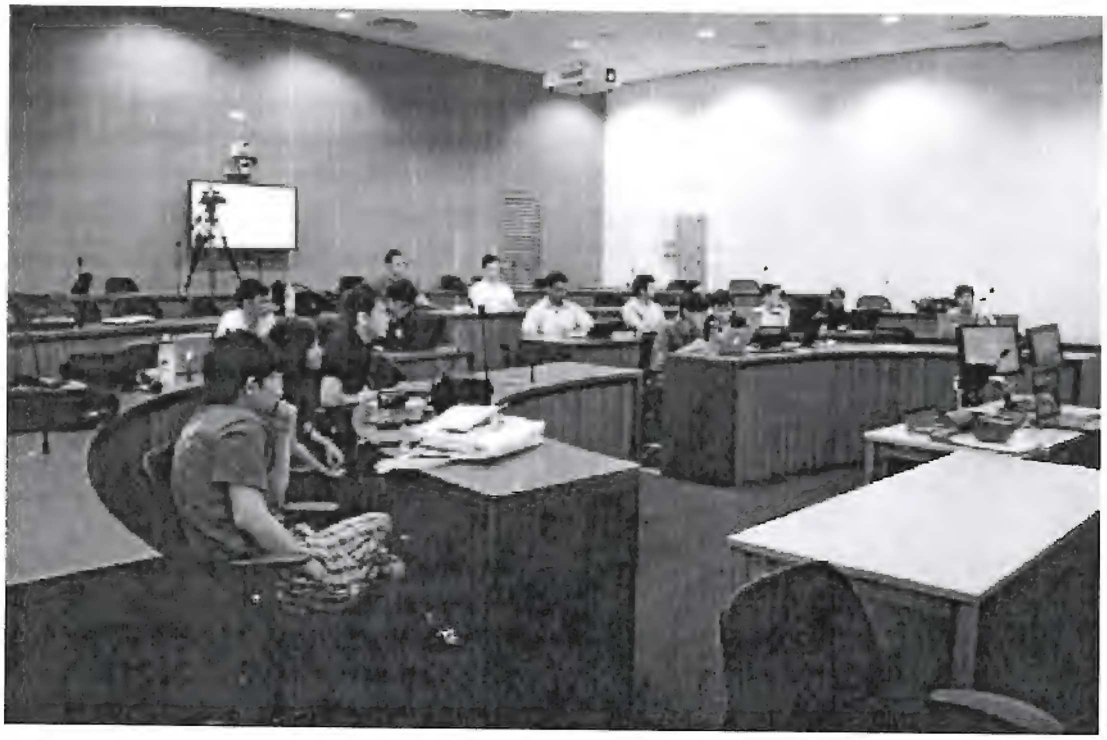
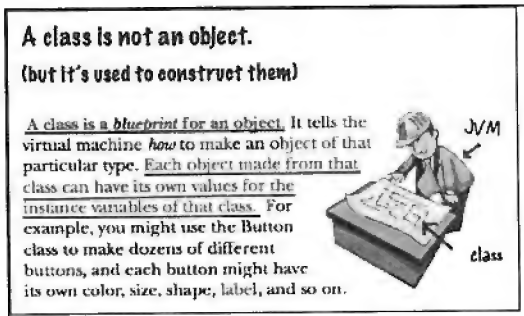


Figure 5. Experimental group setting

*Pre-reading Materials.* The students are given a set of pre-reading material one day before the actual lessons. Each set consists of topics that are related to the next day's lesson. From pre-reading, the students are expected to develop an overview of the topics before attending the class on the following day. These pre-readings provide background knowledge to the students in order to actively participate in the peer-to-peer interactive discussions. The pre-reading materials also serve as a reference library when students have questions or meet obstacles during the collaborative lab sessions and integrated designettes. As a result, the pre-reading material is designed to have simple presentation, comprehensive information and understandable language.

The sets of pre-reading material are developed based on the concept inventory. We composed these topics based on related daily interactive discussions, initially led by student peer-to-peer teams. Figure 6 show exemplar pre-reading materials based on selected course inventory topics, Figure 7 shows exemplar practical exercises included as part of the pre-reading materials, and Figure 8 shows sample codes as part of the pre-reading materials that express OOP concepts.

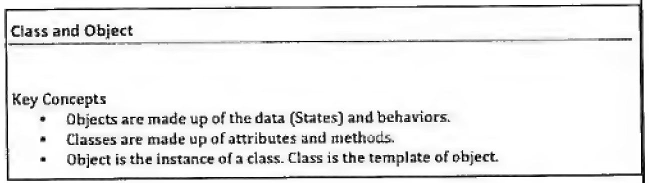


**A class is not an object.**

**(but it's used to construct them)**

A class is a *blueprint* for an object. It tells the virtual machine *how* to make an object of that particular type. Each object made from that class can have its own values for the instance variables of that class. For example, you might use the Button class to make dozens of different buttons, and each button might have its own color, size, shape, label, and so on.

(a) Underlined important OOP concepts

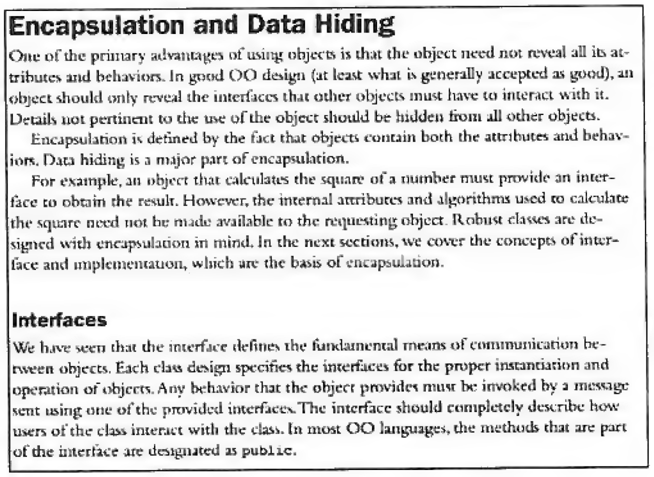


Class and Object

Key Concepts

- Objects are made up of the data (States) and behaviors.
- Classes are made up of attributes and methods.
- Object is the instance of a class. Class is the template of object.

(b) Summaries of key concepts integrated in the peer-to-peer pre-reading materials



**Encapsulation and Data Hiding**

One of the primary advantages of using objects is that the object need not reveal all its attributes and behaviors. In good OO design (at least what is generally accepted as good), an object should only reveal the interfaces that other objects must have to interact with it. Details not pertinent to the use of the object should be hidden from all other objects.

Encapsulation is defined by the fact that objects contain both the attributes and behaviors. Data hiding is a major part of encapsulation.

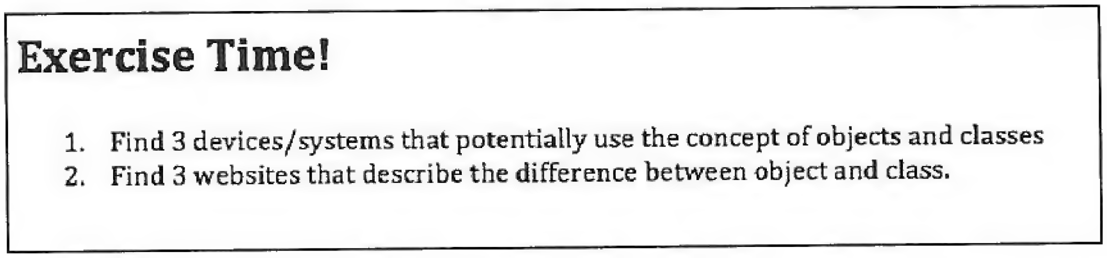
For example, an object that calculates the square of a number must provide an interface to obtain the result. However, the internal attributes and algorithms used to calculate the square need not be made available to the requesting object. Robust classes are designed with encapsulation in mind. In the next sections, we cover the concepts of interface and implementation, which are the basis of encapsulation.

**Interfaces**

We have seen that the interface defines the fundamental means of communication between objects. Each class design specifies the interfaces for the proper instantiation and operation of objects. Any behavior that the object provides must be invoked by a message sent using one of the provided interfaces. The interface should completely describe how users of the class interact with the class. In most OO languages, the methods that are part of the interface are designated as `public`.

(c) Theoretical explanations

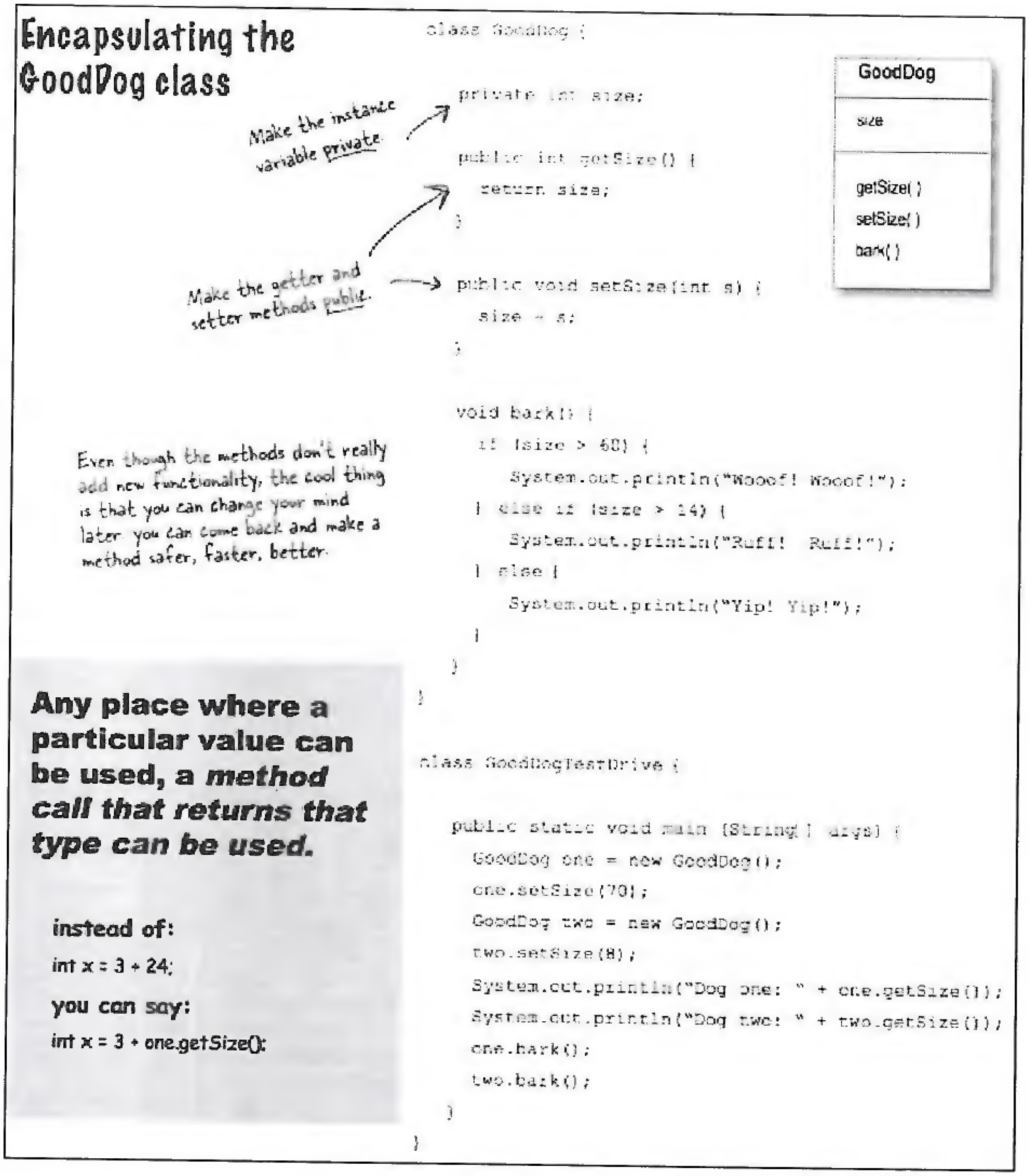
Figure 6. Exemplar excerpts of pre-reading materials (Sierra, 2005; Weisfeld, 2009)



**Exercise Time!**

1. Find 3 devices/systems that potentially use the concept of objects and classes
2. Find 3 websites that describe the difference between object and class.

Figure 7. Pre-reading materials: examples of embedded, practical exercises



Encapsulating the GoodDog class

Make the instance variable private.

Make the getter and setter methods public.

Even though the methods don't really add new functionality, the cool thing is that you can change your mind later. you can come back and make a method safer, faster, better.

```
class GoodDog {

   private int size;

   public int getSize() {
     return size;
   }

   public void setSize(int s) {
     size = s;
   }

   void bark() {
     if (size > 60) {
        System.out.println("Wooof! Wooof!");
     } else if (size > 14) {
        System.out.println("Ruff!  Ruff!");
     } else {
        System.out.println("Yip! Yip!");
     }
   }
}

class GoodDogTestDrive {

   public static void main (String[] args) {
     GoodDog one = new GoodDog();
     one.setSize(70);
     GoodDog two = new GoodDog();
     two.setSize(8);
     System.out.println("Dog one: " + one.getSize());
     System.out.println("Dog two: " + two.getSize());
     one.bark();
     two.bark();
   }
}
```

| GoodDog |
|---|
| size |
| getSize( )<br>setSize( )<br>bark( ) |

**Any place where a particular value can be used, a *method call that returns that type can be used.***

**instead of:**

int x = 3 + 24;

**you can say:**

int x = 3 + one.getSize();

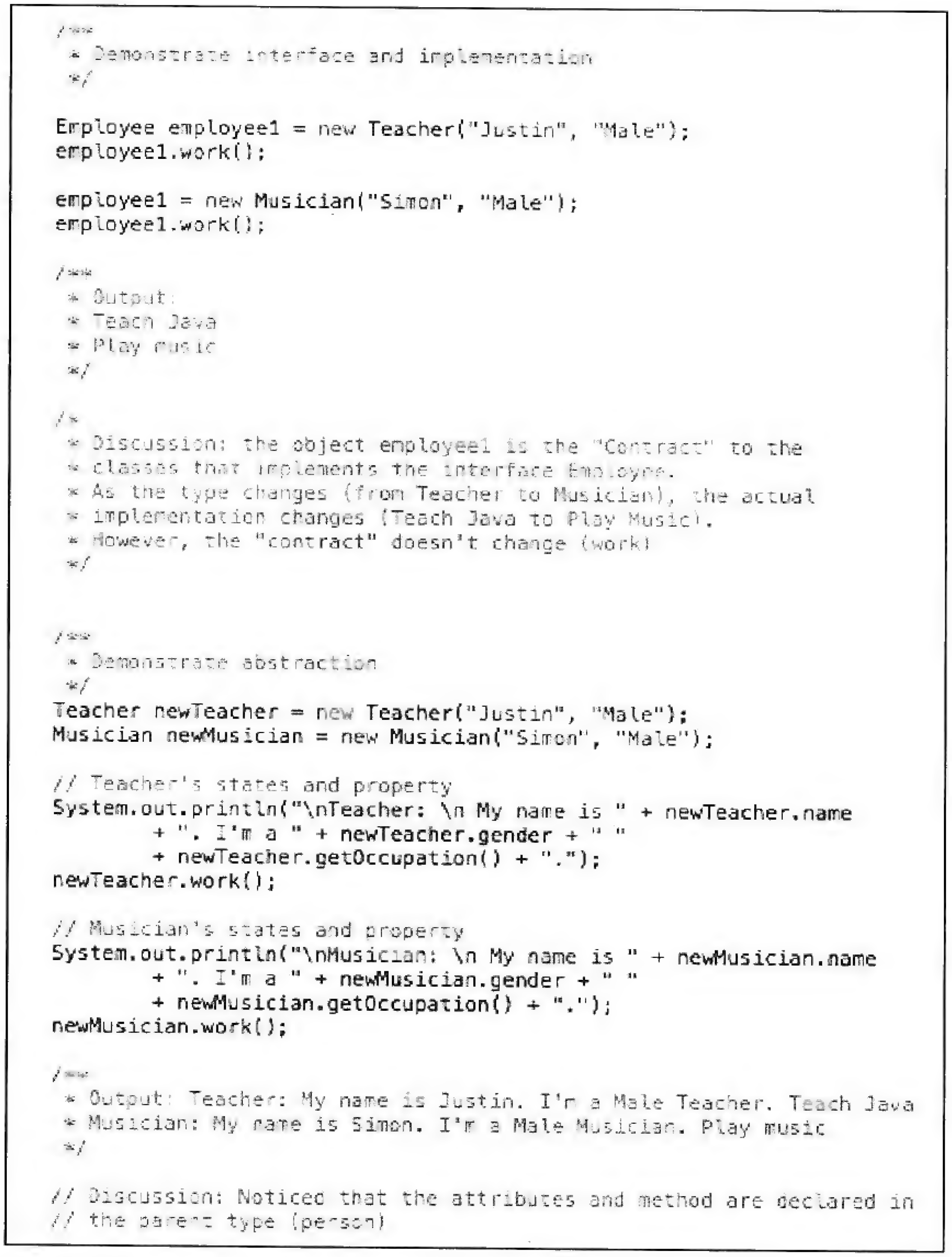
Figure 8. Pre-reading materials: sample codes expressing OOP concepts (Sierra, 2005)

***Peer-to-Peer Interactive Discussions***. On the first day, the first interactive discussion section pairs the students as two-person teams. The two-person teams interact with real-world devices and creations, i.e., show-n-tell, relating the topics of the course to everyday life. The instructor leads the discussion on how programming / algorithms play a role in devices and creations with the groups for 15 minutes. The devices and creations include phones, mobile apps, electronic devices and web programs on the Internet.

The second interactive discussion section is the daily topic discussion. It consists of two parts. During the first part, the instructor spends 15 minutes introducing the software design process in general according to personal experience. Subsequently, the instructor introduces a design problem and its settings (designette). Collaboratively with the class, the instructor decomposes the design problem into five modules. These five modules are to be used in the Lab sections everyday – two modules on the first two days and one on the third day.

The second part of the topic discussion is a peer-to-peer interactive discussion on the first days' course content: Object Oriented Concepts. In the first 30 minutes, students share what they learn from the pre-reading materials in groups and to the class (Fig. 9). In the next 45 minutes, the instructor presents a personal understanding on the same set of material with the aid of PowerPoint slides, with *ad hoc* examples, that are programmed interactively and collaboratively with the students (Fig. 10).

Figure 9. Peer-to-Peer interactive discussion and class presentation



```
/**
 * Demonstrate interface and implementation
 */

Employee employee1 = new Teacher("Justin", "Male");
employee1.work();

employee1 = new Musician("Simon", "Male");
employee1.work();

/**
 * Output:
 * Teach Java
 * Play music
 */

/*
 * Discussion: the object employee1 is the "Contract" to the
 * classes that implements the interface Employee.
 * As the type changes (from Teacher to Musician), the actual
 * implementation changes (Teach Java to Play Music).
 * However, the "contract" doesn't change (work)
 */



/**
 * Demonstrate abstraction
 */
Teacher newTeacher = new Teacher("Justin", "Male");
Musician newMusician = new Musician("Simon", "Male");

// Teacher's states and property
System.out.println("\nTeacher: \n My name is " + newTeacher.name
        + ". I'm a " + newTeacher.gender + " "
        + newTeacher.getOccupation() + ".");
newTeacher.work();

// Musician's states and property
System.out.println("\nMusician: \n My name is " + newMusician.name
        + ". I'm a " + newMusician.gender + " "
        + newMusician.getOccupation() + ".");
newMusician.work();

/**
 * Output: Teacher: My name is Justin. I'm a Male Teacher. Teach Java
 * Musician: My name is Simon. I'm a Male Musician. Play music
 */

// Discussion: Noticed that the attributes and method are declared in
// the parent type (person)
```

Figure 10. Instructor sample code developed and used during the interactive discussion sessions

***Collaborative DBL Lab***. The collaborative lab sessions are driven by a design problem, i.e., design-based learning via designettes (Wood, 2012). As an example designette and collaborative lab experience, students work in groups of two persons to develop a Fab Lab Management System to manage booking, training and schedule viewing of various equipment, processes and facilities as part of a university prototyping shop. Below is a detailed description of the designette provided to the students as part of the experimental study.

Design Problem - Fab Lab Management System (FLMS):

A software system is needed with the following characteristics. All characteristics described below must be included in the Fab Lab Management software system.

System Description:

**The Fab Lab Management System (FLMS)** is to be designed as part of the lab safety authentication. It automates the process of authenticating students' access to the lab and its machines and processes. In order to gain the access, students will need to use FLMS to complete a safety test. By obtaining the safety certificate, students are then able to use the other features.

The usage or each machine or process in the lab is also restricted by licenses. Students will have to attend training sessions and complete the certificate tests to obtain various licenses in order to use the respective machines. Using FLMS, students can view the training schedule and make appointments subjected to availability.

Once the licenses are obtained, students can activate the respective machines by scanning the student identification card on the card reader attached the machine.

Key Features:

**1. System Architecture:** The system architecture of FLMS allows effective communication among various entities such as user, certificate, and appointment. Each of these entities has concrete entities, such as student and admin are derived entities of user, and safety certificate and various machine licenses are entities of certificate.

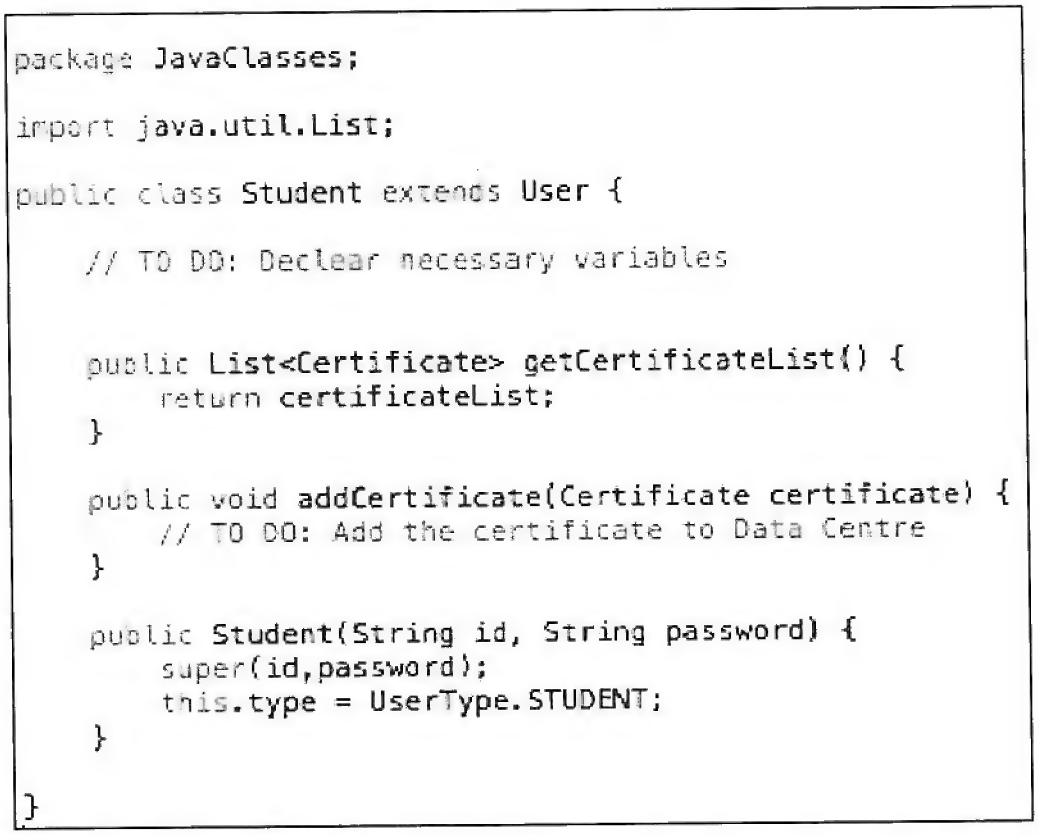
**2. Log in:** The system requires users to log in before using any other features. The users (student/admin) log in with their id and password. The system is able to distinguish between students and admin, and direct them to respective pages.

**3. Certificate Test:** The system allows students to check and book slots for training and examination of the certificates. Upon passing a test, students will obtain the respective license. With the license, students can then activate the machine by scanning the student identification card on the card reader attached to each machine. In this process, the student's id number will be passed to the system to check if the license is already obtained to operate the machine.

**4. Simulation on Machine Activation:** Using the identification number and password, students can activate the machines in the Fab Lab. During authentication, the system checks if the student has obtained the license of the machine. Once the authentication passes, the students will be given 30 minutes to run the machine.

**5. Machine Booking:** The system also allows the student to check the calendar for the availability of the machines, and book a time slot to use the available machines. (More descriptions may be written. These may be used for parallel programming topic.)

**6. Data Centre:** The system has an entity to store the data which is shared among all other entities. Common Data such as user info, machine info and machine schedules will be stored in the data center. The data center does not allow any other entity to create a copy or instantiate it at any point of time. The data will not be erased by system restart.

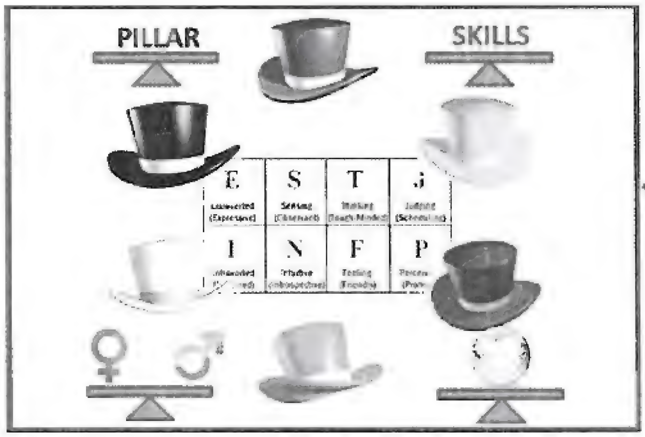


```
package JavaClasses;

import java.util.List;

public class Student extends User {

    // TO DO: Declear necessary variables


    public List<Certificate> getCertificateList() {
        return certificateList;
    }

    public void addCertificate(Certificate certificate) {
        // TO DO: Add the certificate to Data Centre
    }

    public Student(String id, String password) {
        super(id,password);
        this.type = UserType.STUDENT;
    }

}
```

Figure 11. Example pseudo-code shared with student peer-to-peer teams as part of collaborative labs

There are in total five lab sessions, two sessions each on the first two days and one on the last day. For each lab session, students work on one module of the design problem. The level of complexity and difficulty increases gradually. In Collaborative Lab 1, students are to complete the first module of the design problem, which is discussed in the morning. Students are given the basic algorithm, a portion of pseudo-code (Fig. 11) and a test case. In order to complete the problem, it requires the students to complete missing portions of provided code and run the test case. Collaborative Lab 2 entails work on the second module of the designette. The instructor introduces and describes an algorithm. Students develop the code for the complete code for the algorithm and run the given test case to validate the code. In Collaborative Lab 3, students are given an algorithm. They are expected to write the complete code and test case to solve the problem (design module) and validate the code. Collaborative Lab 4 entails work on the fourth module of the design problem. Students are to design the algorithm to address the problem. Students then develop complete code for the module and write the test case to validate the algorithm and code. Collaborative Lab 5 entails work on the fifth module of the designette. Students are to design the algorithm to address the problem. Subsequently, students develop two alternative codes for the algorithm. Lastly, students write the test cases to validate the algorithm and codes.

***Final Epitome Assignment***. On the afternoon of the third day, both the control and experimental groups are given a designette project as the final assignment. A real customer is invited to present the design problem separately in both classes (Fig. 12.2). The problem corresponds to a real situation where students of an institution need to be sorted into "balanced" teams (according to a given set of requirements) to improve their expected results and performance. The presentation includes background knowledge, design requirements and expected outcome. The students are expected to design an algorithm and associated software for effective design group formation based on a set of requirements, including the team members' personality type (Jensen, 2000), working style, skill set, gender and nationality. The resulting software is intended to be used to systematically create design teams of a predetermined size range for university courses of co-curricular activities, as shown in Fig. 12.1.



| # | GENDER | NATIONALITY | MBTI ASSESTMENT | | | | | | | | 6 HATS ASSESTMENT | | | | | | PILLAR | SKILLS | TEAM # |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Male/Female | | E | I | N | S | T | F | P | J | W | Y | R | G | Blue | B | EPD/ESD/ASD/ISTD | Yes/No | |
| 1 | M | Col | 66 | 22 | 11 | | | | | | 22 | 11 | | | | | EPD | Y | |
| 2 | F | Sing | | 33 | | | | | | | 33 | | | | | | ASD | N | |
| 3 | F | Thai | | | 33 | | | | | | | 33 | | | | | ISTD | N | |
| 4 | M | UK | | | | 66 | | 88 | | | | | 66 | | | | ESD | N | |
| 5 | M | USA | | | 44 | | 55 | | | | | 44 | 22 | 11 | | | EPD | Y | |
| 6 | M | Mex | | | | | | | | | | | 33 | | | | ESD | N | |
| 7 | F | Jap | | | | | | 12 | | | | | | 33 | | | EPD | Y | |
| 8 | M | Chn | | | | | | | | | | | | | 66 | | ASD | Y | |
| 9 | F | Mal | | | | | | | | | | | | 44 | | 55 | ISTD | N | |

Figure 12.1. Input file, requirements and expected output

The workshop students receive a file with the information recorded in the table shown in Fig 1, and resulting software is intended to systematically create design teams (blue header column) of a predetermined size range (5-6) for university courses of co-curricular activities.

Figure 12.2. Dr. Diana Moreno presenting a design problem to the control and experimental groups

The students in the control and experimental groups are assigned to design teams of four or five persons to work on the project for one and a half days (half a day on the third day and full day on the fourth day). Teams are formed separately for both classes. During this period, the two instructors are available in answering questions via email. On the last day, the students are brought back together. Each design team presents their solution to the design problem to the customer and instructors (Fig. 13).

Upon completion of the final epitome design project, all students in the course complete a post-test on the object-oriented programming concepts, which is identical to the pre-test taken at the beginning of the courses. All participants also complete a learning experience self-assessment. These learning assessment and evaluation instruments provide the experimental data for the study, i.e., pre- and post-tests of the course content, a questionnaire on the learning experience by both groups, and key demographic information. These data are analyzed statistically to measure the learning outcomes of key concepts when comparing the control and experimental groups. Self-assessment of learning styles and approaches are also analyzed across the sample sizes for both groups and with respect to the pre-knowledge and demographics of the participants.

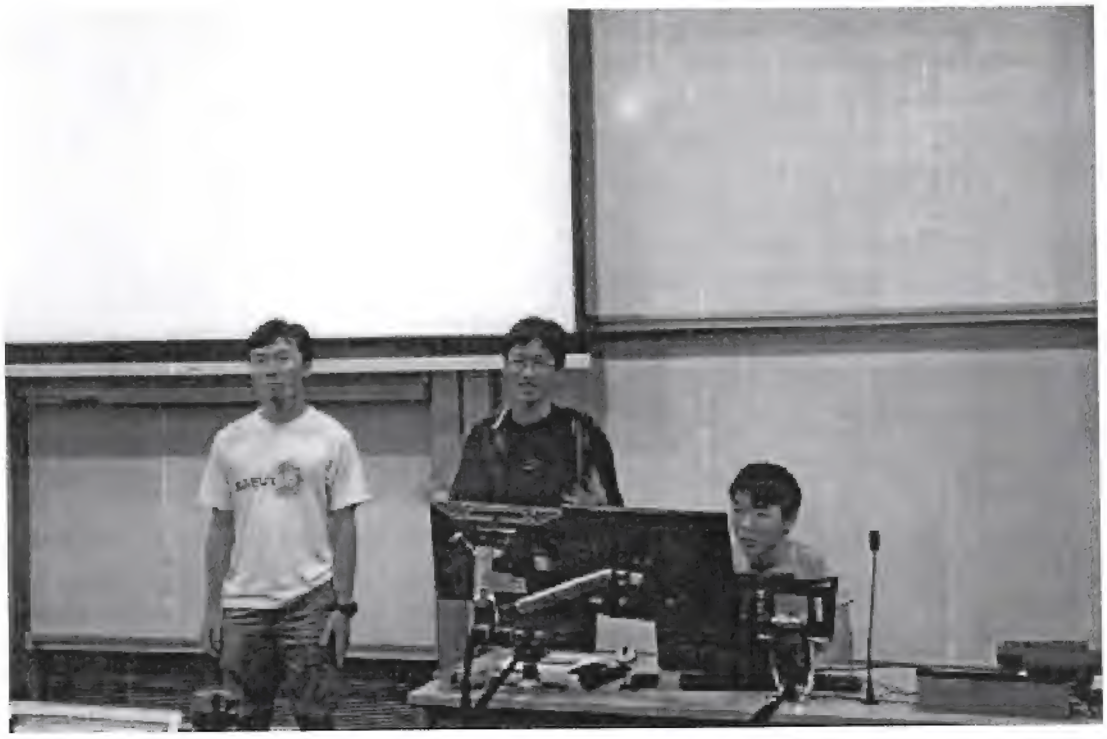


Figure 13. Exemplar student presentation of the epitome design project

## IV. Assessment Instruments, Data Collection, and Analysis

### *IV.1 Learning Experience Self-Assessment – Survey*

The survey (Appendix B) is designed as a key quantitative measurement for the experimental study. It is designed to enable participants to rate their experiences with particular classroom and lab experiences, in addition to the overall course. The self-assessment includes three sections. Section I gathers demographic information of the student such as gender, age, academic studies and programming experience. Section II gathers quantitative rankings on each course component for both the experimental group and control group. Students rank the lecture sessions, interactive discussions, labs, relevance between lessons and labs, and overall experience. The scale is on the following basis: to what extent the components are interesting, motivating, inspiring, engaging and difficult. We likewise gather participant's self-assessment on efforts, focus, successfulness and number of hours students spend outside the class. Section III gathers participants' qualitative feedback and suggestions for course improvement.

### *IV.2 Pre- and Post-Tests*

Pre- and post-tests, and their comparison of results, are a key quantitative measurement of this pedagogical experiment. Both instructors for the experimental study were informed of the key course topics based on the concept inventory. As described above, both the traditional and active-DBL frameworks used were structured to focus on these concepts. However, the pre- and post-tests, designed and constructed from the course inventory, were not disclosed to the instructors until after the course was completed. This process seeks to avoid participants in either course from receiving any hints during lectures or interactive discussions, which may affect the accuracy of the result of the experimental study and associated statistical analysis.

While designing the pre- and post-test, the intent is to ensure the quality of the questions, the ability of the questions to measure the participants' understanding of key concepts, and how well the participants are able to apply the concepts in problem solving. Hence, all questions are set based on the Object-Oriented Programming concept inventory. Both the pre- and post-tests contain 35 multiple-choice questions with only one correct answer per question. In terms of question type, the questions include Theory and Application categories. In terms of difficulty, the assessment instruments consist of Basic Level and Advanced Level questions. There are thus four sub-types of questions, namely Basic Theory, Advanced Theory, Basic Application, and Advanced Application. By comparing both the overall categories and sub-categories, we can assess the participants' learning of key concepts and clearly measure the advantages and disadvantages between the traditional pedagogy and the active-DBL approach.

The pre-test was completed by participants in the briefing before the actual course started. This pre-test serves as an indicator and reference of a participant's programming background and prior knowledge and understanding in OOP. On the other hand, the post-test was completed on the last day of the course, after the students had completed all lessons and assignments. The post-test allows us to measure how well the students have learned the key concepts and provide a comparison with pre-test conditions.

Both the pre- and post-tests are designed to be identical. By doing so, we aim to obtain minimal error in measuring the quality, effectiveness and efficiency of both classes. Figure 15 shows pre- and post-test sample questions, and Appendix A provides a complete pre-/post-test as a concept inventory. The overall difficulty of the assessment instruments is judged to be relatively high. Each participant student is given 45 minutes to complete the test. In order to choose the correct answer, the students must fully understand the tested concepts. In addition, every question has 4 – 8 options with only one correct answer. This design aims to avoid students from scoring through guessing.

| | |
|---|---|
| Basic Theory (BT) | What is the appropriate data type for this value: "Dog"?<br>A. int<br>B. String<br>C. double<br>D. boolean<br>E. I don't know |
| Advanced Theory (AT) | Which of the following statement is not one of the restrictions in each enum declaration?<br>A. Enum constants are implicitly final, because they declare constants that shouldn't be modified.<br>B. Enum constants are implicitly static.<br>C. Any attempt to create an object of an enum type with operator "new" results in a compilation error.<br>D. The keyword static must be present while declaring the fields of an enum.<br>E. I don't know. |
| Basic Application (BA) | Which of the following piece of code will compile?<br>A. ArrayList<Number> numberList = new ArrayList<Integer>();<br>B. ArrayList<? Extends Number> numberList = new ArrayList<Integer>();<br>C. ArrayList<? super Number> numberList = new ArrayList<Integer>();<br>D. ArrayList<Number> numberList = Arrays.asList(1,2,3,4,5);<br>E. None of the above.<br>F. I don't know. |
| Advanced Application (AA) | `public class MyClass1 {`<br>`  public void m1(int i) {}`<br>`  public void m2(int i) {}`<br>`  public static void m3(int i) {}`<br>`  public static void m4(int i) {}`<br>`}`<br>`public class MyClass2 extends MyClass1 {`<br>`  public static void m1(int i) {}`<br>`  public void m2(int i) {}`<br>`  public void m3(int i) {}`<br>`  public static void m4(int i) {}`<br>`}`<br><br>Which method hides a method in the superclass?<br>A. m1<br>B. m2<br>C. m3<br>D. m4<br>E. All of the above<br>F. None of the above<br>G. I don't know |

Figure 15. Pre-/Post-Test sample questions

## *IV.3 Assessment Data and Analysis*

There are two key assessments and evaluations of the experimental study. The first is the learning self-assessment survey. The other is participant performance on the pre- and post-tests.

The survey aims to measure the qualitative self-assessment and self-reported motivation for learning fundamental programming and software engineering principles. All questions expect single answers. For all questions except 6.1, there are five categories of answers, ranging between two extremes. For coding purposes, we map a Likert scale to the responses, corresponding to -2 to 2 to represent the assessment score. Question 6.1 asks participants about the number of hours spent in independent study after class. For this question, we use numbers from 0-8 to represent the answers.

A power analysis for sample size shows a minimum sample size of approximately eight participants per group to obtain a power of 80% for a difference in survey scores between 0.5 and 1.0, and an expected standard deviation of 0.5. Of course, a higher sample size is desired to assure this power level and avoid significant biases that may exist due to such factors as participant background.

Based on the voluntary enrollment during IAP, there are a total of 16 actual participants (out of 29 registered participants) that attended all lessons, completed the survey and both tests. Nine of the participants are in the experimental group, whereas seven are in the control group. The data collected for the experimental study are summarized in Appendix C.

The data is analyzed by comparing the experimental and control groups using a student-t, two-tail test. The average scores and standard deviations are calculated for both the experimental and control groups, followed by the statistical test. Table 2 summarizes the results of this analysis. Two of the questions have clear statistical significance with p-values less than 0.05. One question is in the p-value range of 0.05 to 0.10. And two questions have p-values very near 0.10. All other questions may be used to analyze common trends, when agreement exists in positive or negative Likert scores.

Table 2. Statistical Analysis of Learning Self-Assessment Survey Results

| Question | | Average | | | Compare | | |
|---|---|---|---|---|---|---|---|
| | | Exp | Ctrl | Diff | STD Exp | STD Ctrl | t test |
| Q1.1 | Lecture/Discussion: Uninteresting / Interesting | 1.13 | 1.14 | -0.02 | 0.64 | 0.38 | 0.7686 |
| Q1.2 | Lecture/Discussion: De-Motivating / Motivating | 0.50 | 0.71 | -0.21 | 0.53 | 0.76 | 0.6459 |
| Q1.3 | Lecture/Discussion: Frustrating / Inspiring | 0.63 | 0.71 | -0.09 | 0.52 | 0.95 | 0.9071 |
| Q1.4 | Lecture/Discussion: Unengaging / Engaging | 1.38 | 1.14 | 0.23 | 0.52 | 0.69 | 0.5513 |
| Q1.5 | Lecture/Discussion: Difficult / Easy | 0.13 | 0.00 | 0.13 | 0.64 | 0.82 | 0.4440 |
| Q2.1 | Lab: Uninteresting / Interesting | 1.00 | 0.57 | 0.43 | 0.53 | 0.53 | 0.1265 |
| Q2.2 | Lab: De-motivating / Motivating | 0.63 | 0.43 | 0.20 | 0.52 | 0.79 | 0.5014 |
| Q2.3 | Lab: Frustrating / Inspiring | 0.25 | 0.14 | 0.11 | 0.71 | 0.69 | 0.5970 |
| Q2.4 | Lab: Unengaging / Engaging | 0.88 | -0.14 | 1.02 | 0.35 | 0.90 | 0.0148 |
| Q2.5 | Lab: Difficult / Easy | -0.63 | -0.29 | -0.34 | 0.92 | 0.49 | 0.6874 |
| Q3.1 | Lecture with Lab: Irralevant / Closely relevant | 1.13 | 0.29 | 0.84 | 0.64 | 1.11 | 0.0795 |
| Q3.2 | Lecture with Lab: Unhelpful / Helpful | 1.13 | 0.57 | 0.55 | 0.64 | 0.98 | 0.1612 |
| Q4.1 | Self-accessment: Min effort - max effort | 0.63 | 1.14 | -0.52 | 0.52 | 0.69 | 0.1533 |
| Q4.2 | Self-accessment: Unfocussed - Focus | 0.38 | 1.00 | -0.63 | 0.74 | 0.58 | 0.1104 |
| Q4.3 | Self-accessment: Unsuccessful - Successful | 0.00 | 1.14 | -1.14 | 0.76 | 0.38 | 0.0046 |
| Q5.1 | Time vs Work | 1.00 | 0.71 | 0.29 | 0.00 | 0.49 | 0.4360 |
| Q6.1 | Hr spent outside classroom | 4.63 | 3.57 | 1.05 | 1.92 | 1.27 | 0.5696 |
| Q7.1 | Overall: Uninteresting / Interesting | 1.25 | 1.14 | 0.11 | 0.46 | 0.38 | 0.7045 |
| Q7.2 | Overall: Demotivating / Motivating | 0.75 | 0.86 | -0.11 | 0.46 | 0.90 | 0.6273 |
| Q7.3 | Overall: Frustrating / Inspiring | 0.88 | 0.71 | 0.16 | 0.64 | 0.76 | 0.6264 |
| Q7.4 | Overall: Unengaging / Engaging | 1.00 | 1.00 | 0.00 | 0.53 | 0.58 | 1.0000 |
| Q7.5 | Overall: Difficult / Easy | -0.13 | 0.00 | -0.13 | 0.35 | 0.82 | 0.7877 |

***Pre-/Post-Test.*** In both of the pre- and post-tests, all questions have only one correct answer. In coding the responses, a zero (0) is used to represent a wrong answer and a one (1) is used to represent a correct answer. The pre-test scores and the post-test data are listed in Appendix D. Because of a lower power for sample size the pre- and post-test, and because of no valid statistical significance between the experimental and control groups, only raw data are chosen for the answers to concept inventory questions, in addition to the differences in pre- and post-test performance across participants. The distributions of performance on the pre- and post-tests, listed in the "Sum" rows and summarized as means and standard deviations, are compared across all participants, as shown in Table 3, using a two-tailed student-t test. Statistical significance between the pre- and post-test performance exists, where the p-value is 0.00000132, i.e., << 0.01.

Table 3. Statistical analysis of performance difference across participants

| Mean | | Standard Diviation | | Student-t |
|---|---|---|---|---|
| Pre-test | Post-test | Pre-test | Post-test | |
| 15.69 | 22.44 | 3.24 | 3.10 | 0.00000132 |

## V. Discussion and Conclusions

Section IV presents the assessment and evaluation instruments for the experimental study, the raw data collected with these instruments, and a basic statistical analyses of the results. Considering, first, the participants' self-assessment of learning, the experimental-study results show a range of implications. Based on the survey data, we conclude as follows (referring to Tables 2):

For Interactive Discussion / Lecture sessions: (i) both groups find these sessions of the course capture their interest; (ii) both groups are motivated within these sessions of the course, on average, with a higher variance in the control group; (iii) both groups are inspired, on average, with a higher variance for the control group; (iv) both groups find these sessions to be engaging; and (v) both groups are neutral regarding the level of difficulty, i.e., the level of difficulty is neither too high or too low.

For Lab sessions: (i) the experimental group finds the lab sessions to be more interesting; (ii) both groups are positive about the lab sessions in general; (iii) both groups are relatively positive in finding the lab sessions to be inspiring; (iv) the experimental group has a clearly engaging lab experience, greater than that of the control group; and (v) both groups find the labs to be difficult, with higher variance by the active learning group.

For relevance between Interactive Discussion / Lecture and Lab: (i) both groups are indicate that the sessions show relevant connectivity, with the active learning group having a significantly higher rating and the control group having a higher variation; (ii) both groups find the sessions to be helpful, with the experimental group having higher rating and the control group having higher variation; (iii) both groups devoted significant, but with the control group having higher indicated effort.

For Overall Self-Assessment: (i) both groups exhibited focus, but with higher indicated focus by the control group; (ii) the control group indicates more personal success, where the focus is on homework and homework-like labs as opposed to more open-ended design-based labs; (iii) both groups feel challenged by quantity of work for the given time; (iv) the experimental group, on average, dedicates more time to the sessions, but there does not exist a statistically significant difference; and (v) both groups find the course very interesting; (vi) both groups find the course motivating; (vii) both groups find the course inspiring; (viii) both find the course engaging; and (ix) both groups find the course engaging.

Overall, based on a self-reported assessment, the active learning - DBL group finds the course elements to be more interesting, engaging, and with much higher connectivity than the control group. However, even with these clear distinctions, the control group feels they achieve more personal success than the active-learning group. We interpret that their response may be based on their experience with a more homework-based course content, or, alternatively, active learning was potentially more challenging, especially the design-based laboratories and peer-to-peer interactive discussions. Both groups find the overall experience to be interesting, motivating, inspiring and engaging, indicating the careful design of each pedagogical approach.

In addition to the self-assessment surveys, the pre- and post-test results show a clear finding (Tables 4-6). Even though there exists no statistically significant differences in performance between the experimental and control groups, both groups showed a marked improvement in learning the course content. This result, as measured by a systematically developed concept inventory, is reasonable for a short course with purely volunteer participants. But it also provides credence to the experimental findings. Because of the distinct learning by both groups, the experimental study is not biased by poorly designed course content or course implementation frameworks. Instead, participants learned a range of desired concepts, where the implication is that either course framework could be used in future software engineering courses, but with higher likelihood of successful in student performance for integrated active learning and DBL due to the higher degree of interest, engagement, and connectivity perceived by the participants through various course elements.

***Contributions***. Returning to the stated hypothesis of the experimental study, i.e., ***"There exists significant improvement in the engagement, student interest, and motivation for software engineering content using an integrated approach of active and deign-based learning compared to traditional teaching approaches,"*** the experiment results provide clear positive support for this hypothesis. Thus, a key contribution of this paper is the potential shown for combining active-learning and DBL approaches, especially in adapting and extending recent engineering education research findings, such as designettes. This contribution extends to the particular elements developed for object-oriented programming courses, and, more generally, software engineering. Other significant contributions of the study include the development of an initial concept inventory for object-oriented programming concepts, a self-assessment instrument for student learning, and course frameworks demonstrating a spectrum for teaching engineering subject matter.

***Limitations and Future Work.*** While the results of the experimental study are very encouraging regarding active learning and design-based learning, there are a number of limitations in the current study. These limitations include a relatively low sample size, especially for the pre- and post-test data; evolutions and validation needed for the concept inventory; the use of different instructors to teach courses for the control versus experimental groups; and the use of a short-course format and setting as opposed to full term courses. Future work will address approaches to overcome these limitations and may include a hybrid of teaching approaches, higher sample size studies, factor studies on different types of peer-to-peer discussions and designettes, and refinements of the concept inventory. Moving forward, we will seek opportunity to implement the model in a semester-long course, with students who are not volunteers.

## VI. Acknowledgements

This work is primary supported by a grant from the SUTD-MIT International Design Centre (IDC) (http://www.sutd.edu.sg/idc.aspx). In addition, we acknowledge the dedicated efforts of the participants involved in experimental study, as well as Peter Jun and Shanaka Ransiri, who performed as the role of teaching assistant in the study. Any opinions, findings, or recommendations are those of the authors and do not necessarily reflect the views of the sponsors.

## VII. References

Adams, R.S., Turns, J., Newman, J., & Atman, C. (2004). "An Engineering Education," Unpublished manuscript, University of Washington.

Aglan, H.A., Ali, S.F. (1996). "Hands-on Experiences: An Integral Part of Engineering Curriculum Reform," *Journal of Engineering Education*, pp. 327-330, Oct.

Andersin, L. W., Krathwohl, D. R., et al. (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Addison Wesley Longman.

Anderson, L. & Northwood, D. (2002). "Recruitment and retention programs to increase diversity in engineering," International Conference on Engineering Education Manchester, United Kingdom. Retrieved 6 December 2007 from http://www.ineer.org/Events/ICEE2002/Proceedings/Papers/Index/O065-O070/O069.pdf.

Anwar, S. (2004). "Use of Digital Systems Fundamentals to Teach Design in a Freshman Engineering Design and Graphics Course," *International Conference on Engineering Education*, Gainesville, Florida, October 16-21.

Barr, R., Schmidt, P., Krueger, T., and Twu, C. (2000). An Introduction to Engineering Through an Integrated Reverse Engineering and Design Graphics Project, *Journal of Engineering Education*, 89(4):413-418.

Bean, John C. (2001). Engaging Ideas: The Professor's Guide to Integrating Writing, Critical Thinking, and Active Learning in the Classroom. San Francisco, CA:Wiley.Beaudoin, D. L. and D. F. Ollis. (1995). "A product and process engineering laboratory for freshmen," *ASEE Journal of Engineering Education*, July, pp. 279-284, Vol. 84, No. 3.

Bilén, S. G., Devon, R., & Okudan, G. E. (2002). "Core methods in teaching global product development," *ICEE* 2002, Manchester, 22 August.

Blessing, L. (2002). "What is this thing called design research?" *International CIRP Design Seminar*, Hong Kong, 16-18 May.

Bloch, J. (2008). *Effective java*. Prentice Hall.

Bloom B. S. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*. New York: David McKay Co Inc.

Bloom's Taxonomy Table. (2007). Available at: http://oregonstate.edu/instruct/coursedev/models/id/taxonomy/#table.

Bloom's Taxonomy Figure. (2010). Available at: http://www.thecaepepreschool.com/bloomspop.html, August, 2010.

Bonwell, C.C., "Active Learning and Learning Styles," *Active Learning Workshops Conference*, Content available at http://www.active-learning-site.com/vark.htm.

Borchert, R., Jensen, D., and Yates, D. (1999). "Hands-on and Visualization Modules for Enhancement of Learning in Mechanics: Development and Assessment in the Context of Myers Briggs, Types and VARK Learning Styles," *Proceedings of ASEE Annual Conference*, Charlotte, NC, June.

Bowe, M., Jensen, D., Feland, J., and Self, B. (2000). "When Multimedia Doesn't Work: An Assessment of Visualization Modules for Learning Enhancements in Mechanics," *Proceedings of ASEE Annual Conference*, St. Louis, June.

Bransford, J. D., Brown, A., and Cocking, R.R., eds. (2000). *How People Learn: Brain, Mind, Experience, and School.* Washington, D.C. National Academy Press.

Budd, T., An Introduction to Object-Oriented Programming, Addison-Wesley, Reading, MA, 1991.

Calabro, K.M., Kiger, K.T., Lawson, W., and G.M. Zhang. (2008). "New Directions in Freshman Engineering Design at the University of Maryland," *FIE '08 Conference Proceedings*, pp. T2D-6, October 2008, Saratoga Springs, NY.

Campbell, M. (2002). "Teaching Machine Design through Product Emulation," *ASEE Annual Conference*, Session 2366, Montréal, Quebec, Canada.

Carberry, A.R., Lee, H.-S., and Ohland, M.W. (2010). "Measuring engineering design self-efficacy," *Journal of Engineering Education*. Vol.99 (1), pp. 71-79.

CECI – Computer Engineering Concept Inventory, January 2012, Foundation Coalition, http://www.foundationcoalition.org/home/keycomponents/concept/computer.html.

Charyton, C., and Merrill, J.A. (2009). "Assessing general creativity and creative engineering design in first year engineering students," *Journal of Engineering Education,* Vol.98 (2), pp. 145-156.

Chesler, N., D'Angelo, C., Arastoopour, G., and Shaffer, D.W. (2011). "Use of Professional Practice Simulation in a First-Year Introduction Engineering Course," Paper presented at the *American Society for Engineering Education Conference (ASEE)*, Vancouver, BC.

Costa, P., and McCrae, R. (1992). *NEO PI-R Professional Manual*, Psychological Assessment Resources, Inc., Odessa, FL.

Crawford, R.H., Wood, K.L., Fowler, M., and Norrell, J. (1994). "An Engineering Design Curriculum for the Elementary Grades," *ASEE Journal of Engineering Education*, Vol. 83, No. 2, pp. 172-181.

Dally, J. W. and G. M. Zhang. (1993). "A freshman engineering design course," *J. Eng. Ed*., 82, 2, pp. 83-91.

Dennis, S., Bowe, M., Ball, J., Jensen, D. (2001). "A Student-Developed Teaching Demo of an Automatic Transmission," *Proceedings of the ASEE Annual Conference*, Albuquerque NM, June.

Douglas, J. et al. (2004). Engineering in the K-12 Classroom. An Analysis of Current Practices & Guidelines for the Future. A Production of the ASEE Engineering K12 Center. [http://www.engineeringk12.org/educators/taking_a_closer_look/ documents/Engineering_in_the_K-12_Classroom.pdf]

Dutson, A.J., Todd, R.H., Magleby, S.P., and Sorensen, C.D. (1997). "A Review of Literature on Teaching Design Through Project-Oriented Capstone Courses," *Journal of Engineering Education*, Vol. 76, No. 1, pp. 17–28.

Davis, W. S., Business Systems Analysis and Design, Wadsworth, Belmont, CA, 1994.

Dym, C.L. (1994). "Teaching Design to Freshmen: Style and Content," *Journal of Engineering Education*, Vol. 83, No. 4, pp. 303–310.

Dym, C.L. (1999). "Learning Engineering: Design, Languages, and Experiences," *Journal of Engineering Education*, Vol. 88, No. 2, pp. 145–148.

Dym, C.L. and Little, L. (2003). *Engineering Design: A Project-Based Introduction*, 2nd ed., New York, N.Y.: John Wiley.

Dym, C.L. (2004). "Design, Systems, and Engineering Education," *International Journal of Engineering Education*, Vol. 20, No. 3, pp. 305–312.

Dym, C., Agogino, A., Eris, O., Frey, D., and Leifer, Larry. (2005). "Engineering Design Thinking, Teaching, and Learning," *Journal of Engineering Education*, volume 94, issue 1, pp. 103-120.

Eccles, J.S., Adler, T., Futterman, R., Goff, S., Kaczala, C., Meece, J., & Midgley, C. (1983). Expectancies, values, and academic behaviors. In J. Spence (Ed.), Achievement and Achievement Motives (pp. 75-146). San Francisco: W.H. Freeman.

Eccles, J. S. (1994). "Understanding women's educational and occupational choices — Applying the Eccles et al. model of achievement-related choices," *Psychology of Women Quarterly*, 18, 585–609.

Eccles, J.S. (2007). "Where are all the women? Gender differences in participation in physical science and engineering," In S. J. Ceci & W. M. Williams (Eds.), W*hy aren't more women in science? Top researchers debate the evidence* (pp. 199-212). Washington, DC: American Psychological Association.

Eder, W. E. (1994). "Comparisons – Learning Theories, Design Theory, Science," *Journal of Engineering Education*, pp. 111-119, April.

Eckel, B. (2003). *Thinking in JAVA*. Prentice Hall PTR.

Fagerberg, J., Mowery, D., & Nelson, R. (2004). *The Oxford Handbook of Innovation*. Oxford: Oxford University Press. Felder, R. M., Felder, G. N., Mauney, M., Hamrin, C. E., & Dietz, E. J. (1995). A longitudinal study of engineering student performance and retention. III. Gender differences in student performance and attitudes. *Journal of Engineering Education*, 84, 151–163.

Felder, R.M., and Silverman, L.K. (1988). "Learning and Teaching Styles in Engineering Education," *Journal of Engineering Education*, Vol. 78, No. 7, pp. 674-681.

Felder, R. & Brent, R. (2005). "Understanding student differences," *Journal of Engineering Education*, 94, 57-72.

Green, M. G., Wood, K. L., VanderLeest, S. H., Duda, F. T., Erikson, C., & Van Gaalen, N. (2004, June). Service-Learning Approaches to International Humanitarian Design Projects: A Model Based on Experiences of Faith-Based Institutions. In *Proceedings 2004 American Society for Engineering Education Conference.*

Hadjerrouit, S. (1999, June). A constructivist approach to object-oriented design and programming. In *ACM SIGCSE Bulletin* (Vol. 31, No. 3, pp. 171-174). ACM.

Hake, R. R. (1987). "Promoting Student Crossover to the Newtonian World," *Am. J. Phys.*, 55:878-884.

Hake, R. R. (1992). "Socratic Pedagogy in the Introductory Physics Laboratory," *The Phys. Teach*, 30:546-552.

Hake, R. R. (1998). "Interactive-Engagement vs Traditional Methods: A Six-Thousand Student Survey of Mechanics Test Data for Introductory Physics Courses," *Am. J. Phys*., 66:64-74.

Hanson, James L. (1999). "Early Experimentation with Civil Engineering Materials," *Proceedings of the 1999 ASEE Annual Conference & Exposition*, Jun.

Heller, P., Keith, R., and Anderson, S. (1992a). "Teaching Problem Solving Through Cooperative Grouping. Part I: Group Versus Individual Problem Solving," *Am. J. Phys*., 60:627-636.

Heller, P., Keith, R., and Anderson, S. (1992b). "Teaching Problem Solving Through Cooperative Grouping. Part II: Designing Problems and Structuring Groups," *Am. J. Phys.*, 60:627-636.

Hsi, S., Agogino, A. (1995). "Scaffolding Knowledge Integration through Designing Multimedia Case Studies of Engineering Design," *Proceedings of ASEE Frontiers in Education Conference*, available at http://fie.engrng.pitt.edu/fie95/4d1/4d11/4d11.htm.

Holzer, S., Andruet, R. (2000). Experiential Learning in Mechanics with Multimedia, *Int. Journal of Engr. Education*, Vol. 16. No. 5.

Horvath, I. (2004). A treatise on order in engineering design. Engineering Design, 15, 155-181.Lin, C-C., Grauer, K. & Castro, J. C. (2011). "There is nothing else to do but make films: Urban youth participation at a film and television school," *International Journal of Education & the Arts*, 12 (SI 1.8).

Hubwieser, P., & Mühling, A. (2011, August). What students (should) know about object oriented programming. In *Proceedings of the seventh international workshop on Computing education research* (pp. 77-84). ACM.

*Human Metrics - Jung Myers Briggs Typology. (2005). www.humanmerics.com*. URL.

Jensen, D.D. and K.L. Wood. (2000). "Incorporating Learning Styles to Enhance Mechanical Engineering Curricula by Restructuring Courses, Increasing Hands-on Activities, & Improving Team Dynamics," *ASME Publication and Presentation for the Award for the Most Innovative Curriculum for the Year 2000, Presented at the ASME Annual Conference*. Orlando, FL.

Jensen, D., Wood, K., Crawford, R., and Crowe, K. (2005). *An Evaluation of the DTEACh Robolab Summer Institute for 2004 – Assessment of Instructional and Hands-on Learning Correlated with MBTI types,* Presented at the ASEE Annual Conference.

Jung, C.G. (1971). *Psychological Types, Volume 6 of the Collected Works of C.G. Jung*. Princeton University Press.

Knight, R. D. (2004). *Five Easy Lessons: Strategies for Successful Physics Teaching*, Addison-Wesley, San Francisco, CA.

Kolb, D.A. (1984). *Experimental Learning: Experience as the Source of Learning and Development*, Prentice Hall, Englewood Cliffs, NJ.

Kurfiss, J.G. (1988). *Critical Thinking: Theory, Research, Practice, and Possibilities.* ASHR-ERIC Higher Education Report No. 2. Washington, D.C.: ERIC Clearinghouse on Higher Education and the Association for the Study of Higher Education.

Laws, P. (1997). "Promoting Active Learning Based on Physics Education Research in Introductory Physics Courses," *Am. J. Phys*., 65:13-21.

Linsey, J., Cobb, B., Jensen, D., Wood, K., and Eways, S. (2006). "Methodology and Tools for Developing Hands-on Active Learning Activities," *Proceedings of 2006 American Society for Engineering Education Annual Conference*, Chicago, IL.

Linsey, J., Talley, A., Jensen, D., Wood, K., Schmidt, K, Kuhr, R., and Eways, S. (2007). "From Tootsie Rolls to Composites: Assessing a Spectrum of Active Learning Activities in Engineering Mechanics," *Proceedings of 2007 American Society for Engineering Education Annual Conference*, Honolulu, HI.

Linsey, J., Talley, A., Jensen, D., Wood, K.L. (2008). "PHLIpS for Active Learning," *ASEE Annual Conference*, Pittsburgh, PA, June, Paper No.: AC-2008-1680.

Linsey, J., Talley, A., White, C. K., Jensen, D., and Wood, K. L. (2009). "From Tootsie Rolls to Broken Bones: An Innovative Approach for Active Learning in Mechanics of Materials," *ASEE Journal of Advances in Engineering Education (AEE)*, 2009, Vol. 1, No. 3, pp. 1-23.

Linsey, J., Tseng, I., Fu, K., Cagan, J., Wood, K., and Schunn, C. (2010). "A study of design fixation, its mitigation and perception in engineering design faculty." *ASME Journal of Mechanical.*

Manning, W. (2009). *SCJP Sun Certified Programmer for Java 6 Exam 310-065 Certification Exam Preparation Course in a Book for Passing the SCJP Sun Certified Programmer for... on Your First Try Certification Study Guide*. Emereo Pty Ltd.

Markman, A., Wood, K.L., Linsey, J., Murphy, J., and Laux, J., "Supporting Innovation by Promoting Analogical Reasoning," Rotman Magazine, Rotman School of Management, University of Toronto, Jan 01, 2011. Prod. #: ROT134-PDF-ENG.

Martin, J. and Odell, J. J., Object-Oriented Analysis and Design, Prentice-Hall, Englewood Cliffs, NJ, 1992.

Mayer, R. E. (2002). "Using Illustrations to Promote Constructivist Learning from Science Text," in J. Otero, Leon, J. A. Graesser, A. and Mahwah, N.J., The Psychology of Science Text Comprehension, pp. 333-356.

Mazur, E. (1997). *Peer Instruction: A User's Manual*, Prentice-Hall, Upper Saddle River, NJ.

McCaulley, M.H., (1990). *The MBTI and Individual Pathways in Engineering Design.* Engineering Education, **80**: p. 537-542.

McDermott, L., Schaffer, P., and Somers, M. (1994). "Research as a Guide to Teaching Introductory Mechanics," *Am. J. Phys.*, 62:46-55.

McDermott, L., Schaffer, P., and the Physics Education Group. (1998). *Tutorials in Introductory Physics*, Prentice-Hall, Upper Saddle River, NJ.

McRae, R.R. and P.T., Costa. (1989). "Reinterpreting the Myers-Briggs Type Indicator from the Perspective of the Five-Factor Model of Personality," *Journal of Personality*, 57(1): p. 17-40.

Messick, S. (1989). Meaning and values in test validation: The science and ethics of assessment. *Educational Researcher*, *18*(2), 5-11.

Messick, S. (1994). The interplay of evidence and consequences in the validation of performance assessments. *Educational Researcher*, *23*(2), 13-23.

Meyer, D. G., Krzyzkowski, R.A. (1994). "Experience Using the Video Jockey System for the Instructional Multimedia Delivery," *Proceeding of the ASEE Frontiers in Education Conference*," pp. 262-266, Nov.

Moritz, S. H., & Blank, G. D. (2005). A design-first curriculum for teaching Java in a CS1 course. *ACM SIGCSE Bulletin*, *37*(2), 89-93.

Myers, I.B., and McCaulley, M.H. (1985). *Manual: A Guide to the Development and Use of the Myers-Briggs Type Indicator*, Consulting Psychologists Press, Palo Alto, CA.

*National Academy of Engineering (NAE). (2005). Assessing the Capacity of the U.S. Engineering Research Enterprise.* [http://www.nae.edu/nae/engecocom.nsf/weblinks/MKEZ-68HQMA?OpenDocument], Washington, DC: National Academy Press.

National Research Council. (1991). *Improving Engineering Design: Competitive Advantage*. Washington, DC: National Academy Press.

Newman, D.J. and A.R. Amir. (2001). "Innovative First Year Aerospace Design Course at MIT," *ASEE Journal of Engineering Education,* 90(3): 375-381, July.

O'Brien, T., Bernold, L. and Akroyd, D. (1998). "Myers-Briggs Type Indicator and Academic Achievement in Engineering Education," *International Journal for Engineering Education*. 14(5) P311-315.

Otto, K. and Wood, K.L. (1998). "A Reverse Engineering and Redesign Methodology," *Journal of Research in Engineering Design*, Vol. 10, No. 4, pp. 226-243.

Otto, K. N. and Wood, K. L. (2001). *Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development*, Prentice-Hall, NY.

Ozgur, E. (2004). *Effective Inquiry for Innovative Engineering*. Academic Press.

Paul Deitel, H. D. (2012). *Java - How to program*. Boston, USA: Pearson Education. Inc.

Paul, R.W. (1987). "Dialogical Thinking: Critical Thought Essential to the Acquisition of Rational Knowledge and Passions." In J. B. Baron and R. J. Sternberg (eds.), *Teaching Thinking Skills: Theory and Practice*. New York: Freeman.

Phares, E. J. (1991). *Introduction to psychology*, 3rd. ed., Harper Collins Publishers, New York.

Piket-May, M .J. and J. P. Avery. (1996). "Freshman design projects: a university/community program providing assistive technology devices," *FIE*, Vol. 2, pp.926-929, *26th Annual Frontiers in Education - Vol 2 (FIE'96)*.

Pisano, G. and Shih, W. (2009). "Restoring American Competitiveness," *Harvard Business Review*. July-August. 114-125.

Prince, M. (2004), "Does Active Learning Work? A Review of the Research," *Journal of Engineering Education*, Vol. 93, No. 3, pp. 223-231.

Pryor, M., Tadepalli, S., and Booth, C., "Persistence and extensibility of web-based PSI courses while maintaining a sense of 'presence'," *Proceedings of the 2009 ASEE Annual Conference and Exposition*, Austin, TX, June, 2009.

Roedel, R. , D. Evans, M. Kawski, B. Doak, M. Politano, S. Duerden, M. Green, J. Kelly and D. Linder. (1995). "An integrated, project-based, introductory course in calculus, physics, english, and engineering," *Proceedings ASEE/IEEE Frontiers in Education Conference*, Atlanta, GA.

Ryckman, R. (1982). *Theories of Personality*, 2nd ed., Brooks-Cole, Monterey, CA.

Schmidt, P. and Beaman, J. (2003). "PROCEED: A Department-Wide Curriculum Reform Initiative in Project-Centered Education," *ASEE Annual Conference*, Paper 461.

Schunn, C., Cagan, J., Paulus, P. and Wood, K.L., (2007). "NSF Workshop in Engineering and Science: The Scientific Basis of Individual and Team Innovation and Discovery," NSF 07-25, www.nsf.gov/pubs/2007/nsf0725/nsf0725.pdf, National Science Foundation, March.

Shavinina, L. V., ed. (2003). *The International Handbook of Innovation*. Elsevier Science.

Sheppard, S.D. (1992). "Dissection as a Learning Tool," *Proceedings of the FIE Conference*, Nasvhille, TN, November.

Sheppard, S. D. (1992). "Mechanical dissection: an experience in how things work," *Proceedings of the Engineering Education: Curriculum Innovation & Integration*, Santa Barbara, CA, Jan. 6-10.

Sheppard, S., and Jenison, R. (1997). "Examples of Freshman Design Education." *Int. J. Engr. Ed.*, Vol. 13, No. 4, pp. 248-26.

Sierra, K., & Bates, B. (2005). *Head First Java*. O'Reilly.

Sokoloff, D. and Thornton, R. (1997). "Using Interactive Lecture Demonstrations to Create an Active Learning Environment," The Phys. Teach., 35:340-347.

Stice, J. E. (1987). "Using Kolb's Learning Cycle to Improve Student Learning," *Engineering Education*, pp. 291-296, Feb.

Streveler, R. A., & Smith, K. A. (2006). Conducting rigorous research in engineering education. *Journal of Engineering Education*, *95*(2), 103-105.

Schunn, C., Cagan, J., Paulus, P. and Wood, K.L., (2007). "NSF Workshop in Engineering and Science: The Scientific Basis of Individual and Team Innovation and Discovery," NSF 07-25, www.nsf.gov/pubs/2007/nsf0725/nsf0725.pdf, National Science Foundation, March.

Talley, A., Lindsey, J., Jensen, D., Wood, K., (2007). "Development and Assessment of Active Learning Products to Enhance Engineering Mechanics Courses", Poster Session- Proceedings of ASEE Annual Conference, Honolulu, HI , June.

Tavakoli, M. S. and J. Mariappan. (2000). "Concurrent teaching of engineering design, analysis and manufacturing," *Int. J. Mech. Eng. Educ.*, 28(4), pp. 321-333.

Ulrich, K. T., & Eppinger, S. D. (2004). *Product Design and Development*, 3rd Ed. New York: McGraw-Hill.

Van Heuvelen, A. and Maloney, D. (1999). "Playing Physics Jeopardy," Am. J. Phys., 67:252-256.

Webster, D. and Kruglanski, A. (1994). "Individual Differences in Need for Cognitive Closure," *Journal of Personality and Social Psychology*, Vol. 67, No. 6, 1049-1062.

Weisfeld, M. (2009). *The object-oriented thought process*. Sams.

Welch, R., Klosky, J.L. (2007). An Online Database and User Community For Physical Models in the Engineering Classroom, *Advances in Engineering Education*, Vol. 1, No. 1, Fall.

White, C.K. (2011), "Taking HEED: Intersections of Women's Lives with Humanitarian Engineering Experiences and Design," Dissertation, Columbia University, Teachers College.

Wilde, D. (2000). "Design Team Formation Using Jungian Typology," *Proceedings National Collegiate Inventors and Innovators Alliance*.

Winblad, A. L., et al., Object Oriented Software, Addison-Wesley, Reading, MA, 1990.

Wood, J., and Wood, K.L. (2000). "The Tinkerer's Pendulum for Machine System's Education: Creating a Basic Hands-On Environment with Mechanical Breadboards," *Proceedings of the ASEE Annual Conference*, St. Louis, MO, June.

Wood, J., Winebrener, D., Bartolomei, J., Jensen, D., Rhymer, D. (2002). "Creating a Visually Rich, Active Learning Environment for Teaching Mechanics of Materials," *Proceedings of the ASEE Annual Conference*, June.

Wood, J., Campbell, M., Wood, K.L., and Jensen, D. (2005). "Enhancing Machine Design by Creating a Basic Hands-On Environment with Mechanical Breadboards," *International Journal of Mechanical Engineering Education*, Vol. 33, No. 1, January, pp. 1-25.

Wood, K., Jensen, D., Bezdek, J., Otto, K. (2001). "Reverse Engineering and Redesign: Courses to Incrementally and Systematically Teach Design," *Journal of Engineering Education*, Vol. 90, No. 3, pp. 363-374, July.

Wood, K. L., Rajesh Elara, M., Kaijima, S., Dritsas, S., Frey, D., White, C. K., Crawford, R. H., Moreno, D., and Pey, K-L. (2012). "A Symphony of Designiettes – Exploring the Boundaries of Design Thinking in Engineering Education," *ASEE Annual Conference*, San Antonio, TX.

Zhu, H., & Zhou, M. (2003, October). Methodology first and language second: a way to teach object-oriented programming. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (pp. 140-147). ACM.

## APPENDIX A – PRE-/POST-TEST CONCEPT INVENTORY

The questions are categorized into Basic/Advanced, and Theory/Application. The four cross-categorizations are Basic Theory (BT), Basic Application (BA), Advanced Theory (AT) and Advanced Application (AA). The questions are in randomized order. The categorizations are not disclosed to the students.

1. [BT] Choose the best definition of a Class.
   A. A special type of object
   B. A group definition, containing the data and function elements necessary to create an object
   C. An action for a program
   D. A group of students in a room
   E. I don't know

2. [BT] Choose the best definition of an object
   A. A thing
   B. An instance of a class
   C. Something you wear
   D. A class
   E. I don't know

3. [BT] What is the appropriate data type for this value: "Dog"
   A. int
   B. String
   C. double
   D. boolean
   E. I don't know

4. [BT] Which of the following item(s) can be considered object(s)?
   A. A circle
   B. A dog

C. A string of text
D. All of the above
E. I don't know

5. [BT] Which of the following statements best describes the relationship between an object and a class?
   A. The term Object is used to describe the memory allocation whereas class is used to describe the data structure. Both of them refer to an identical concept in Object-Oriented Programming.
   B. An object is a template of a class and a class is an instance of an object. The state and behaviors of an object define the attributes and methods of a class respectively.
   C. An object is a template of a class and a class is an instance of an object. The attributes and methods of an object define the state and behaviors of a class respectively.
   D. A class is a template of an object and an object is an instance of a class. The state and behaviors of an object define the attributes and methods of a class respectively.
   E. A class is a template of an object and an object is an instance of a class. The attributes and methods of a class define the state and behaviors of an object respectively.
   F. I don't know

6. [AA] In Addendum 1, what is the output?
   A. Nike 10 0 1 2 3 4 5 6 7 8 9
   B. Nike 11 0 1 2 3 4 5 6 7 8 9
   C. Nike 10 0 10 20 30 40 50 60 70 80 90
   D. Adidas 10 0 1 2 3 4 5 6 7 8 9
   E. Adidas 11 0 1 2 3 4 5 6 7 8 9
   F. Adidas 10 0 10 20 30 40 50 60 70 80 90
   G. Compilation error
   H. I don't know

7. [BT] What is the role of a constructor?
   A. Make the code confusing
   B. Create an instance of a class
   C. Create names for methods
   D. Create some type of change in the state of an object
   E. I don't know

8. [BA] In Addendum 1, which of the classes practice(s) encapsulation?
   A. Ball
   B. Basketball
   C. Soccer
   D. All of the above
   E. None of the above
   F. I don't know

9. [BT] Which of the following item(s) can be considered object(s)?
   A. A boolean
   B. A dog
   C. An arraylist of Integers
   D. All of the above
   E. Two of the above
   F. None of the above
   G. I don't know

10. [AT] What is essential for encapsulation?
    A. It exposes of the data in order to allow efficient access of the data. The common practice of encapsulation is to mark instance variables public and mark getters and setters private.

B. It exposes of the data in order to allow efficient access of the data. The common practice of encapsulation is to mark instance variables private and mark getters and setters public.
C. It restricts access to the data in order to protect and prevent unnecessary modification of the data. The common practice of encapsulation is to mark instance variables public and mark getters and setters private.
D. It restricts access to the data in order to protect and prevent unnecessary modification of the data. The common practice of encapsulation is to mark instance variables private and mark getters and setters public.
E. I don't know

11. [BT] Which of the following statement are <u>not</u> true?
    A. Encapsulation is a form of data hiding.
    B. Encapsulation typically allows programs to run faster
    C. Encapsulation helps to protect data from corruption
    D. Encapsulation allows for changes to the internal design of a class while the public interface remains unchanged.
    E. Encapsulation usually increases the size of the code.
    F. I don't know

12. [BT] Inheritance allows code
    A. Reusability
    B. Reliability
    C. Usefulness
    D. Correctness
    E. Readability
    F. I don't know

13. [BT] A java subclass can be defined by using the keyword:
    A. public
    B. private
    C. final
    D. extends
    E. inherits
    F. None of the above
    G. I don't know

14. [AA] Refer to Addendum 2, which method overrides a method in the superclass?
    A. m1
    B. m2
    C. m3
    D. m4
    E. All of the above
    F. None of the above
    G. I don't know

15. [AA] Refer to Addendum 2, which method hides a method in the superclass?
    A. m1
    B. m2
    C. m3
    D. m4
    E. All of the above
    F. None of the above
    G. I don't know

16. [BA] Refer to Addendum 1, which of the following statement is true?

A. Ball is a sub-class of Sport
B. Basketball is a sub-class of Ball
C. Soccer is a sub-class of Basketball
D. BallGame is a sub-class of Sport
E. I don't know

17. [BT] Java provides an approach known as _______ as a convenient alternative to implement multiple inheritance:
    A. Sandbox
    B. Inheritance
    C. Packages
    D. Interface
    E. Class
    F. None of the above
    G. I don't know

18. [AT] Which of the following statement is not true?
    A. Polymorphism allows the reference type to be the super class of the actual object
    B. Polymorphism allows you to introduce new sub class types without changing the original code
    C. Polymorphism is a prerequisite of an inheritance hierarchy
    D. With Polymorphism, anything that extends the declared reference variable types can be assigned to the reference variable
    E. None of the above
    F. I don't know

19. [AA] Refer to Addendum 1, which of the following statement will not compile?
    A. Ball ball = new Ball("MyBrand");
    B. Basketball basketball = new Basketball("MyBrand");
    C. Soccer soccer = new Soccer("MyBrand");
    D. Ball ball = new Basketball ("MyBrand");
    E. Sport sport = new Basketball("MyBrand");
    F. All of the above
    G. None of the above
    H. I don't know

20. [AT] Where is Collection interface from?
    A. java.io
    B. java.lang
    C. java.util
    D. java.awt
    E. java.sql
    F. I don't know

21. [BT] A container can
    A. Hold water
    B. Hold objects
    C. Change the behaviors of a reference type
    D. I don't know

22. [BT] Which of the following type belongs to a sub type of Collection?
    A. An array of strings
    B. ArrayList<String>
    C. HashMap<String, String>
    D. Hashtable
    E. All of the above

F. None of the above
G. I don't know

23. [AT] Consider the following statements:
    - Each element must be unique.
    - Duplicate elements must not replace old elements.
    - Elements are not key/value pairs.
    - Accessing an element can be almost as fast as performing a similar operation on an array.

Which of these classes provides the specified features?

A. LinkedList
B. TreeMap
C. TreeSet
D. HashMap
E. HashSet
F. LinkedHashMap
G. Hashtable
H. None of the above
I. I don't know

24. [AT] Which of the following statement is true?
    A. Set allows only one copy of each object.
    B. Set has an auto-assigned index for every element.
    C. Elements in a TreeSet are sorted in descending order.
    D. Elements in a HashSet is sorted in ascending order.
    E. None of the above.
    F. I don't know.

25. [BT] Which of the following code practices generics?
    A. String[] s1 = {"Hello", "World"};
    B. public static void main(String[] args){}
    C. ArrayList<Ineger> intList = new ArrayList<Integer>;
    D. All of the above
    E. I don't know

26. [BT] Which of the following statement is true?
    A. The use of generics allows you to specify the exact types of objects that a particular data structure will store.
    B. The presence of generics provides the means to create general models of methods and classes that can be declared once, but used with many different data types.
    C. The generic collections are backward compatible with Java code that was written before generics were introduced.
    D. To work with generics, every element of the array must be an object of a class or interface type.
    E. All of the above.
    F. I don't know

27. [BA] Which of the following piece of code will compile?
    A. ArrayList<Number> numberList = new ArrayList<Integer>();
    B. ArrayList<? Extends Number> numberList = new ArrayList<Integer>();
    C. ArrayList<? super Number> numberList = new ArrayList<Integer>();
    D. ArrayList<Number> numberList = Arrays.asList(1,2,3,4,5);
    E. None of the above.

F. I don't know.

28. [BT] Enum is:
    A. A special word
    B. A special int
    C. A special String
    D. A special method
    E. A special Class
    F. I don't know

29. [AT] Which of the following statement is true?
    A. Enum defines a set of constants represented as unique identifiers.
    B. An enum type is a reference type.
    C. Enum allows declaration of methods, constructors and fields.
    D. Enum can have a main method.
    E. All of the above.
    F. None of the above.
    G. I don't know.

30. [AT] Which of the following statement is not one of the restrictions in each enum declaration?
    A. Enum constants are implicitly final, because they declare constants that shouldn't be modified.
    B. Enum constants are implicitly static.
    C. Any attempt to create an object of an enum type with operator "new" results in a compilation error.
    D. The keyword static must be present while declaring the fields of an enum.
    E. I don't know.

31. [AP] Refer to Addendum 3. What is the result of compiling and running the following code?
    A. "It is a color type" once followed by "BLUE".
    B. "It is a color type" twice followed by "BLUE".
    C. "It is a color type" thrice followed by " BLUE".
    D. "It is a color type" four times followed by "BLUE".
    E. Compilation fails.
    F. I don't know.

32. [BT] IO in Java stands for
    A. Inside and Outside
    B. Input and Output
    C. Industrial and Organizational
    D. Interest Only
    E. All of the above
    F. I don't know

33. [AT] What is required to make a class serializable?
    A. Encapsulate all attributes
    B. Make all methods public or protected
    C. Implement Serializable interface
    D. Explicitly declare SerialVersionUID
    E. All of the above
    F. I don't know

34. [AT] Which of the following statements describes a feature of Java's io?
    A. Ability to export objects with their states and behaviors into a plain text file
    B. Provide a uniform set of methods that accept and return File objects

C. Comparability
D. All of the above
E. I don't know

35. [BT] Which of the following statements is true about XML
    A. XML was designed to transport and store data.
    B. XML tags are pre-defined
    C. XML is not human-readable
    D. XML can only be read by applications that are developed using object-oriented programming language.
    E. All of the above
    F. None of the above
    G. I don't know

## Addendum 1

```
public interface Sport {void play();}
public abstract class Ball implements Sport {
        private String brand;
        public String getBrand() {return brand;}
        public void setBrand(String brand) {this.brand = brand;}
        public Ball(String brand) {this.brand = brand;}
        public void play() {System.out.println("Start playing!");}
}
public class Basketball extends Ball {
        public Basketball(String brand) {super(brand);}
        public void work() {System.out.println("Throw!");}
}
public class Soccer extends Ball {
        public Soccer(String brand) {super(brand); }
        public void play() {System.out.println("Kick!"); }
}
public class BallGame {
        public static void main(String[] args) {
                Ball ballGame = new Basketball("Nike");
                int numberOfPlayers = 10;
                int[] jerseyNumbers = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
                changeStats(ballGame, numberOfPlayers, jerseyNumbers);
System.out.print(ballGame.getBrand() + " " + numberOfPlayers + " ");
                for(int i : jerseyNumbers){System.out.print(i + " ");}
        }
        private static void changeStats(Ball firstGame, int numberOfPlayers,
                        int[] jerseyNumbers) {
                firstGame = new Soccer("Adidas");
                numberOfPlayers = 11;
                for (int i = 0; i<jerseyNumbers.length;i++){jerseyNumbers[i]*=10;}
        }
}
```

**Addendum 2**

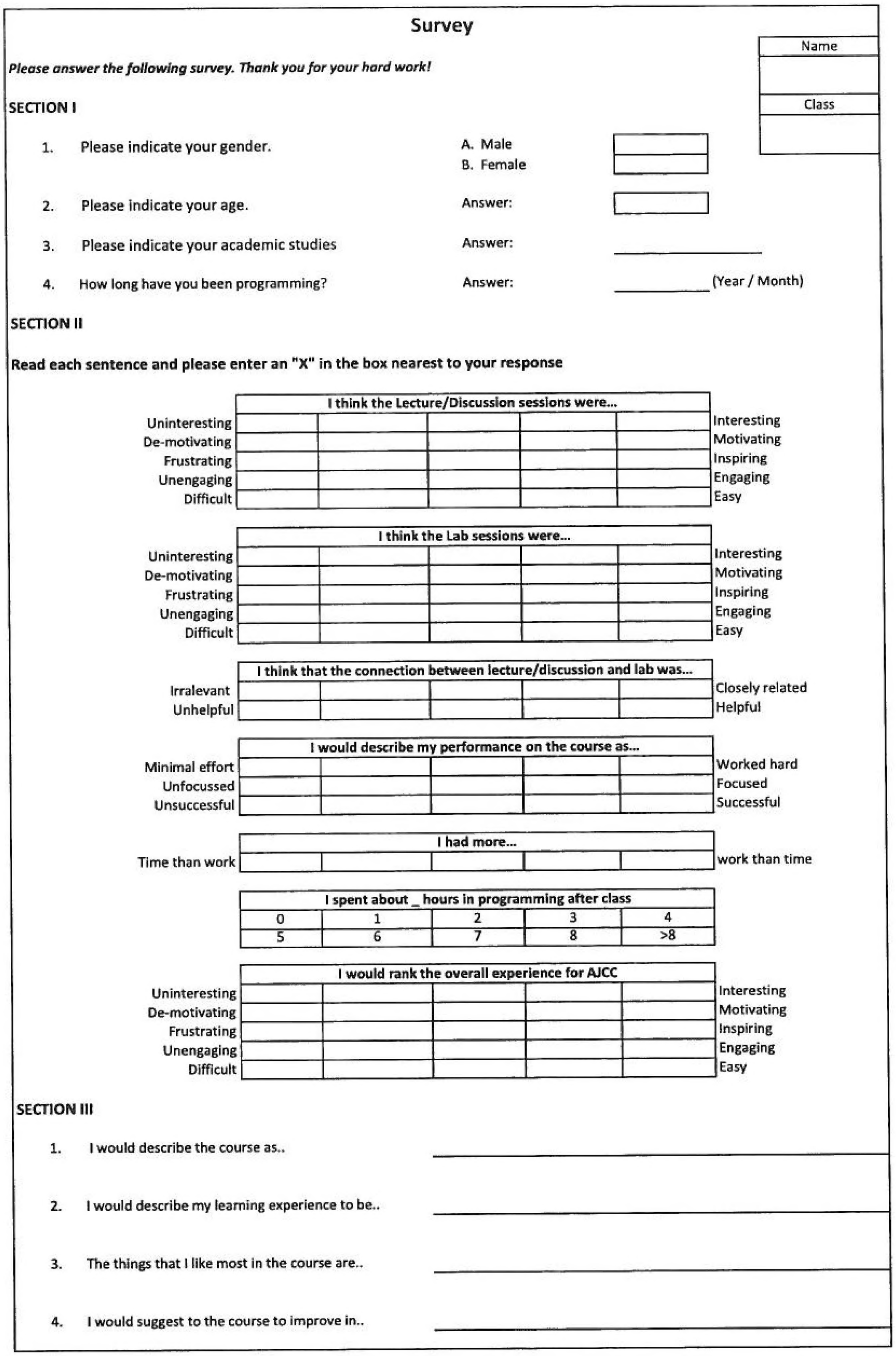
```
public class MyClass1 {
  public void m1(int i) {}
  public void m2(int i) {}
  public static void m3(int i) {}
  public static void m4(int i) {}
}
public class MyClass2 extends MyClass1 {
  public static void m1(int i) {}
  public void m2(int i) {}
  public void m3(int i) {}
  public static void m4(int i) {}
}
```

**Addendum 3**

```
enum Color {
        RED, BLUE, GREEN;
        private AccountType() {System.out.println("It is a color type");}
}
class TestEnum {
        public static void main(String[] args) {
                System.out.println(Color.BLUE);
        }
}
```

## Appendix B – Survey

The survey form serves as the learning self-assessment instrument for the experimental study.



### Survey

| |
|---|
| Name |
| |
| Class |
| |

*Please answer the following survey. Thank you for your hard work!*

**SECTION I**

1. Please indicate your gender. A. Male ☐ B. Female ☐
2. Please indicate your age. Answer: ☐
3. Please indicate your academic studies Answer: ____________
4. How long have you been programming? Answer: __________ (Year / Month)

**SECTION II**

**Read each sentence and please enter an "X" in the box nearest to your response**

| | I think the Lecture/Discussion sessions were... | | | | | |
|---|---|---|---|---|---|---|
| Uninteresting | | | | | | Interesting |
| De-motivating | | | | | | Motivating |
| Frustrating | | | | | | Inspiring |
| Unengaging | | | | | | Engaging |
| Difficult | | | | | | Easy |

| | I think the Lab sessions were... | | | | | |
|---|---|---|---|---|---|---|
| Uninteresting | | | | | | Interesting |
| De-motivating | | | | | | Motivating |
| Frustrating | | | | | | Inspiring |
| Unengaging | | | | | | Engaging |
| Difficult | | | | | | Easy |

| | I think that the connection between lecture/discussion and lab was... | | | | | |
|---|---|---|---|---|---|---|
| Irralevant | | | | | | Closely related |
| Unhelpful | | | | | | Helpful |

| | I would describe my performance on the course as... | | | | | |
|---|---|---|---|---|---|---|
| Minimal effort | | | | | | Worked hard |
| Unfocussed | | | | | | Focused |
| Unsuccessful | | | | | | Successful |

| | I had more... | | | | | |
|---|---|---|---|---|---|---|
| Time than work | | | | | | work than time |

| I spent about _ hours in programming after class | | | | |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | >8 |

| | I would rank the overall experience for AJCC | | | | | |
|---|---|---|---|---|---|---|
| Uninteresting | | | | | | Interesting |
| De-motivating | | | | | | Motivating |
| Frustrating | | | | | | Inspiring |
| Unengaging | | | | | | Engaging |
| Difficult | | | | | | Easy |

**SECTION III**

1. I would describe the course as.. ____________
2. I would describe my learning experience to be.. ____________
3. The things that I like most in the course are.. ____________
4. I would suggest to the course to improve in.. ____________

## Appendix C: Learning Self-Assessment Survey Result

| Question | Content | Experimental Group | | | | | | | | | Control Group | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | C1 | C2 | C3 | C4 | C5 | C6 | C7 |
| Q1.1 | Lecture/Discussion: Uninteresting / Interesting | 1 | 2 | 1 | 0 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 1 |
| Q1.2 | Lecture/Discussion: De-Motivating / Motivating | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 |
| Q1.3 | Lecture/Discussion: Frustrating / Inspiring | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 2 | 2 | 0 | 0 | 1 | 0 |
| Q1.4 | Lecture/Discussion: Unengaging / Engaging | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 0 | 1 | 2 | 1 | 1 | 2 | 1 |
| Q1.5 | Lecture/Discussion: Difficult / Easy | 0 | 0 | 0 | 0 | 1 | 0 | 1 | -1 | 2 | 0 | 1 | 0 | -1 | 0 | 1 | -1 |
| Q2.1 | Lab: Uninteresting / Interesting | 1 | 1 | 0 | 1 | 2 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| Q2.2 | Lab: De-motivating / Motivating | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 2 | 1 | 0 | 0 | 0 | 0 |
| Q2.3 | Lab: Frustrating / Inspiring | 0 | 1 | 1 | 1 | 0 | 0 | 0 | -1 | 1 | 0 | 1 | 1 | 0 | -1 | 0 | 0 |
| Q2.4 | Lab: Unengaging / Engaging | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | -1 | -1 | 0 | -1 | 0 | 1 | 1 |
| Q2.5 | Lab: Difficult / Easy | 0 | 0 | -2 | -1 | 1 | -1 | -1 | -1 | 1 | 0 | 0 | 0 | 0 | -1 | 0 | -1 |
| Q3.1 | Lecture with Lab: Irralevant / Closely relevant | 1 | 1 | 0 | 1 | 2 | 1 | 2 | 1 | 2 | -1 | -1 | 0 | 1 | 1 | 0 | 2 |
| Q3.2 | Lecture with Lab: Unhelpful / Helpful | 1 | 1 | 1 | 0 | 2 | 1 | 2 | 1 | 2 | -1 | 1 | 0 | 1 | 1 | 0 | 2 |
| Q4.1 | Self-accessment: Min effort - max effort | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 0 |
| Q4.2 | Self-accessment: Unfocussed - Focus | 0 | -1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 1 |
| Q4.3 | Self-accessment: Unsuccessful - Successful | 0 | 0 | -1 | 1 | 1 | 0 | -1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 |
| Q5.1 | Time vs Work | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| Q6.1 | Hr spent outside classroom | 2 | 4 | 4 | 6 | 4 | 8 | 6 | 3 | 0 | 5 | 4 | 2 | 5 | 3 | 4 | 2 |
| Q7.1 | Overall: Uninteresting / Interesting | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 |
| Q7.2 | Overall: Demotivating / Motivating | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 0 | 2 | 1 |
| Q7.3 | Overall: Frustrating / Inspiring | 1 | 1 | 1 | 0 | 1 | 0 | 2 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 2 | 0 |
| Q7.4 | Overall: Unengaging / Engaging | 1 | 1 | 1 | 0 | 1 | 1 | 2 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 2 | 1 |
| Q7.5 | Overall: Difficult / Easy | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1 | -1 | 0 | 0 | -1 | 1 |

## Appendix D: Pre- / Post- Test data

Table 1. Pre-test scores across participants

| Pre-test | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Qn** | **P1** | **P2** | **P3** | **P4** | **P5** | **P6** | **P7** | **P8** | **P9** | **P10** | **P11** | **P12** | **P13** | **P14** | **P15** | **P16** |
| **1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **2** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **3** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **4** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| **5** | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| **6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7** | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| **8** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **9** | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| **10** | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| **11** | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| **12** | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| **13** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| **14** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| **15** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **16** | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| **17** | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| **18** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **19** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **20** | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **21** | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| **22** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| **23** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **24** | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **25** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **26** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **27** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **28** | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| **29** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| **30** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| **31** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **32** | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **33** | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **34** | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| **35** | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| **SUM** | 20 | 14 | 14 | 17 | 14 | 22 | 16 | 12 | 15 | 18 | 18 | 13 | 19 | 11 | 17 | 11 |

Table 2. Post-test scores across participants; Improvement over Pre-test

| Post-test | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Qn** | **P1** | **P2** | **P3** | **P4** | **P5** | **P6** | **P7** | **P8** | **P9** | **P10** | **P11** | **P12** | **P13** | **P14** | **P15** | **P16** |
| **1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **2** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **3** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **4** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **5** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| **6** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| **8** | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| **9** | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| **10** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| **11** | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| **12** | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| **13** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **14** | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **15** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **16** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| **17** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| **18** | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **19** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **20** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| **21** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| **22** | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **23** | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| **24** | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| **25** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| **26** | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| **27** | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| **28** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| **29** | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| **30** | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| **31** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **32** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **33** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **34** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| **35** | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| **SUM** | 25 | 24 | 25 | 24 | 25 | 27 | 23 | 21 | 17 | 23 | 27 | 21 | 19 | 18 | 20 | 20 |
| **Improvement:** **Sum(Post-Test) - Sum(Pre-Test)** | 5 | 10 | 11 | 7 | 11 | 5 | 7 | 9 | 2 | 5 | 9 | 8 | 0 | 7 | 3 | 9 |